

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Дисциплина: «Языки программирования»

Отчет по лабораторной работе №16

Работа с данными формата JSON в языке Python

Выполнил студент группы ИТС-б-о-21-1

Усиков Данил

«__»_____20__г.

Подпись студента_____

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Роман Александрович

(подпись)

Ставрополь, 2022

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

Ссылка на репозиторий <https://github.com/danilusikov0913/YPlr6>

Ход работы:

1. Пример

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

Решение: введем следующие команды для работы с файлом формата JSON в интерактивном режиме:

- load - загрузить данные из файла, имя файла должно отделяться от команды load пробелом. Например: load data.json
- save - сохранить сделанные изменения в файл, имя файла должно отделяться от команды save пробелом. Например: save data.json

Напишем программу для решения поставленной задачи.

```
#!/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(
            "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()

```

Рисунок 1 – Код примера

```

>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|   1 | Усиков Д.А              |      Студент        |   2020   |
+-----+-----+-----+-----+

>>> save 2.txt
>>>
>>> list
Список работников пуст.
>>> load 2.txt
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|   1 | Усиков Д.А              |      Студент        |   2020   |
+-----+-----+-----+-----+
>>> |

```

Рисунок 2 – Результат работы

2. Индивидуальное задание 1

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.

```

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

import sys
import json

# Название магазина (фильтр 2х)
# Название товара
# Стоимость
spisok_new = []

def table():

```

```

line = '+-{}-+-{}-+-{}-+-{}-+'.format(
    '-' * 6,
    '-' * 20,
    '-' * 30,
    '-' * 20
)
return line

def table_name():
    post = '| {:^6} | {:^20} | {:^30} | {:^20} | '.format(
        "№",
        "Название магазина",
        "Название товара",
        "Стоимость, руб."
    )
    return post

def table_name_fil(names):
    post = []
    for idx_new, spisok_new_new in enumerate(names, 1):
        post.append(
            '| {:>6} | {:<20} | {:<30} | {:<20} | '.format(
                idx_new,
                spisok_new_new.get('name_shop', ''),
                spisok_new_new.get('name_product', ''),
                spisok_new_new.get('prise', '')
            )
        )
    return post

def save_list_shop(file_name, staff):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_list_shop(file_name):
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main():
    list_shop = []

    while True:
        command = input('>>> ').lower()

        if command == 'exit':
            break

        elif command == 'add':
            name_shop = input('Название магазина: ')
            name_product = input('Название товара: ')
            prise = input('Стоимость товара: ')

            list_shop_new = {
                'name_shop': name_shop,
                'name_product': name_product,
                'prise': prise
            }

            list_shop.append(list_shop_new)

```

```

        if len(list_shop) > 1:
            list_shop.sort(key=lambda item: item.get('name_shop', ''))

    elif command == 'list':
        print(table())
        print(table_name())
        print(table())
        for item_n in table_name_fil(list_shop):
            print(item_n)
        print(table())

    elif command == 'product':
        shop_sear = input('Введите название товара: ')
        search_shop = []
        for shop_sear_itme in list_shop:
            if shop_sear == shop_sear_itme['name_product']:
                search_shop.append(shop_sear_itme)

        if len(search_shop) > 0:
            print(table())
            print(table_name())
            print(table())
            for item_f in table_name_fil(search_shop):
                print(item_f)
            print(table())
        else:
            print('Такого товара не найдено', file=sys.stderr)

    elif command.startswith("save "):
        parts = command.split(maxsplit=1)
        file_name = parts[1]
        save_list_shop(file_name, list_shop)

    elif command.startswith("load "):
        parts = command.split(maxsplit=1)
        file_name = parts[1]
        list_shop = load_list_shop(file_name)

    elif command == 'help':
        print('Список команд:\n')
        print('add - добавить магазин.')
        print('list - вывести список магазинов.')
        print('product <Название> - запросить информацию о товаре.')
        print('help - Справочник.')
        print("load <Название файла без скобок> - загрузить данные из файла;")
        print("save <Название файла без скобок> - сохранить данные в файл;")
        print('exit - Завершить работу программы.')
    else:
        print(f'Команда <{command}> не существует.', file=sys.stderr)
        print('Введите <help> для просмотра доступных команд')

if __name__ == '__main__':
    main()

```

Рисунок 3 – Код индивидуального задания

```
>>> list
```

№	Название магазина	Название товара	Стоимость, руб.
1	Магнит	Молоко	72

```
>>> save 1.txt
>>> list
```

№	Название магазина	Название товара	Стоимость, руб.
---	-------------------	-----------------	-----------------

```
>>> load 1.txt
>>> list
```

№	Название магазина	Название товара	Стоимость, руб.
1	Магнит	Молоко	72

```
>>>
```

Рисунок 4 – Результат работы

3. 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.

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

import sys
import json
import jsonschema

# Название магазина (фильтр 2x)
# Название товара
# Стоимость
spisok_new = []

def table():
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 6,
        '-' * 20,
        '-' * 30,
        '-' * 20
    )
    return line

def table_name():
```



```

post = '| {:^6} | {:^20} | {:^30} | {:^20} | '.format(
    "№",
    "Название магазина",
    "Название товара",
    "Стоимость, руб."
)
return post

def table_name_fil(names):
    post = []
    for idx_new, spisok_new_new in enumerate(names, 1):
        post.append(
            '| {:>6} | {:<20} | {:<30} | {:<20} | '.format(
                idx_new,
                spisok_new_new.get('name_shop', ''),
                spisok_new_new.get('name_product', ''),
                spisok_new_new.get('prise', '')
            )
        )
    return post

def save_list_shop(file_name, staff):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_list_shop(file_name):
    schema = {
        "type": "array",
        "items": [
            {
                "type": "object",
                "properties": {
                    "name_shop": {
                        "type": "string"
                    },
                    "name_product": {
                        "type": "string"
                    },
                    "prise": {
                        "type": {
                            "number"
                        }
                    }
                }
            },
            "required": [
                "name_shop",
                "name_product",
                "prise"
            ]
        ]
    }

    with open(file_name, "r", encoding="utf-8") as fin:
        loadfile = json.load(fin)
        validator = jsonschema.Draft7Validator(schema)
        try:
            if not validator.validate(loadfile):
                print("Валидация прошла успешно")
        except jsonschema.exceptions.ValidationError:
            print("Ошибка валидации", list(validator.iter_errors(loadfile)))

```

```

        return loadfile

def main():
    list_shop = []

    while True:
        command = input('>>> ').lower()

        if command == 'exit':
            break

        elif command == 'add':
            name_shop = input('Название магазина: ')
            name_product = input('Название товара: ')
            prise = int(input('Стоимость товара: '))

            list_shop_new = {
                'name_shop': name_shop,
                'name_product': name_product,
                'prise': prise
            }

            list_shop.append(list_shop_new)

            if len(list_shop) > 1:
                list_shop.sort(key=lambda item: item.get('name_shop', ''))

        elif command == 'list':
            print(table())
            print(table_name())
            print(table())
            for item_n in table_name_fil(list_shop):
                print(item_n)
            print(table())

        elif command == 'product':
            shop_sear = input('Введите название товара: ')
            search_shop = []
            for shop_sear_itme in list_shop:
                if shop_sear == shop_sear_itme['name_product']:
                    search_shop.append(shop_sear_itme)

            if len(search_shop) > 0:
                print(table())
                print(table_name())
                print(table())
                for item_f in table_name_fil(search_shop):
                    print(item_f)
                print(table())
            else:
                print('Такого товара не найдено', file=sys.stderr)

        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_list_shop(file_name, list_shop)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            list_shop = load_list_shop(file_name)

        elif command == 'help':

```

```

print('Список команд:\n')
print('add - добавить магазин.')
print('list - вывести список магазинов.')
print('product <Название> - запросить информацию о товаре.')
print('help - Справочник.')
print("load - загрузить данные из файла;")
print("save - сохранить данные в файл;")
print('exit - Завершить работу программы.')
else:
    print(f'Команда <{command}> не существует.', file=sys.stderr)
    print('Введите <help> для просмотра доступных команд')

if __name__ == '__main__':
    main()

```

Рисунок 5 – Код задания повышенной сложности

```

>>> list
+-----+-----+-----+-----+
|  №   | Название магазина |      Название товара      |  Стоимость, руб.  |
+-----+-----+-----+-----+
>>> load 1.txt
Валидация прошла успешно
>>> list
+-----+-----+-----+-----+
|  №   | Название магазина |      Название товара      |  Стоимость, руб.  |
+-----+-----+-----+-----+
|    1 | Магнит           |      Молоко               |      72           |
+-----+-----+-----+-----+
>>>

```

Рисунок 4 – Результат работы

Контрольные вопросы:

1. Для чего используется JSON?

JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента. Это информативное руководство поможет вам быстрее разобраться с данными, которые вы можете использовать с JSON и основной структурой с синтаксисом этого же формата.

2. Какие типы значений используются в JSON?

Запись, массив, число, литералы, строка

3. Как организована работа со сложными данными в JSON?

4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?

JSON5 — предложенное расширение формата json в соответствии с синтаксисом ECMAScript 5, вызванное тем, что json используется не только для общения между программами, но и создаётся/редактируется вручную. Файл JSON5 всегда является корректным кодом ECMAScript 5. JSON5 обратно совместим с JSON

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5?

JSON5 расширяет формат обмена данными JSON, чтобы сделать его немного более удобным в качестве языка конфигурации:

- Комментарии в стиле JavaScript (как однострочные, так и многострочные) являются законными.
- Ключи объектов могут быть без кавычек, если они являются законными идентификаторами ECMAScript
- Объекты и массивы могут заканчиваться запятыми.
- Строки могут заключаться в одинарные кавычки, и допускаются многострочные строковые литералы.

6. Какие средства предоставляет язык Python для сериализации данных в формате JSON?

Модуль **json** предоставляет удобный метод `dump()` для записи данных в файл. Существует также метод `dumps()` для записи данных в обычную строку. Типы данных Python кодируются в формат JSON в соответствии с интуитивно понятными правилами преобразования

7. В чем отличие функций `json.dump()` и `json.dumps()`?

`dump` отличается от `dumps` тем, что `dump` записывает объект Python в файл JSON, а `dumps` сериализует объект Python и хранит его в виде строки.

8. Какие средства предоставляет язык Python для десериализации данных из формата JSON?

В модуле **json** определены методы `load()` и `loads()`, предназначенные для преобразования кодированных в формате JSON данных в объекты Python.

Подобно операции *сериализации*, также существует таблица преобразования типов, определяющая правила для обратного *декодирования* данных.

9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кириллицу?

Параметр `ensure_ascii`

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