

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего  
образования  
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»**

**Кафедра  
инфокоммуникаций  
Институт цифрового  
развития**

**ОТЧЁТ**  
**по лабораторной работе №2.16**  
Дисциплина: «Основы программной инженерии»  
Тема: «Работа с данными формата JSON в языке Python»

Выполнила:  
студентка 2 курса  
группы Пиж-б-о-21-1  
Джолдошова Мээрим  
Бекболотовна

Ставрополь 2023

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

1. Создайте проект PyCharm в папке репозитория.

Имя	Дата изменения	Тип
.git	23.03.2023 13:27	Папка
PyCharm	23.03.2023 14:01	Папка
.gitignore	09.02.2023 20:56	Файл "
LICENSE	23.03.2023 13:26	Файл

Рисунок 1 – Была создана папка PyCharm

2. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

example1.py	23.03.2023 14:29	JetBrai
jfile.json	23.03.2023 14:37	Исход

Рисунок 2 – Был проработан пример из лабораторной работы

```
Run: main x example1 x
C:\OPI\lbrtt_2.16\PyCharm\Scripts\python.exe C:\OPI\lbrtt_2.16\PyCharm\example1.py
>>> add
Фамилия и инициалы? Castillo R.W
Должность? worker
Год поступления? 2003
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Castillo R.W          |      worker          |      2003      |
+-----+-----+-----+-----+
>>> save jfile.json
>>>
```

Рисунок 3 – Результат работы программы

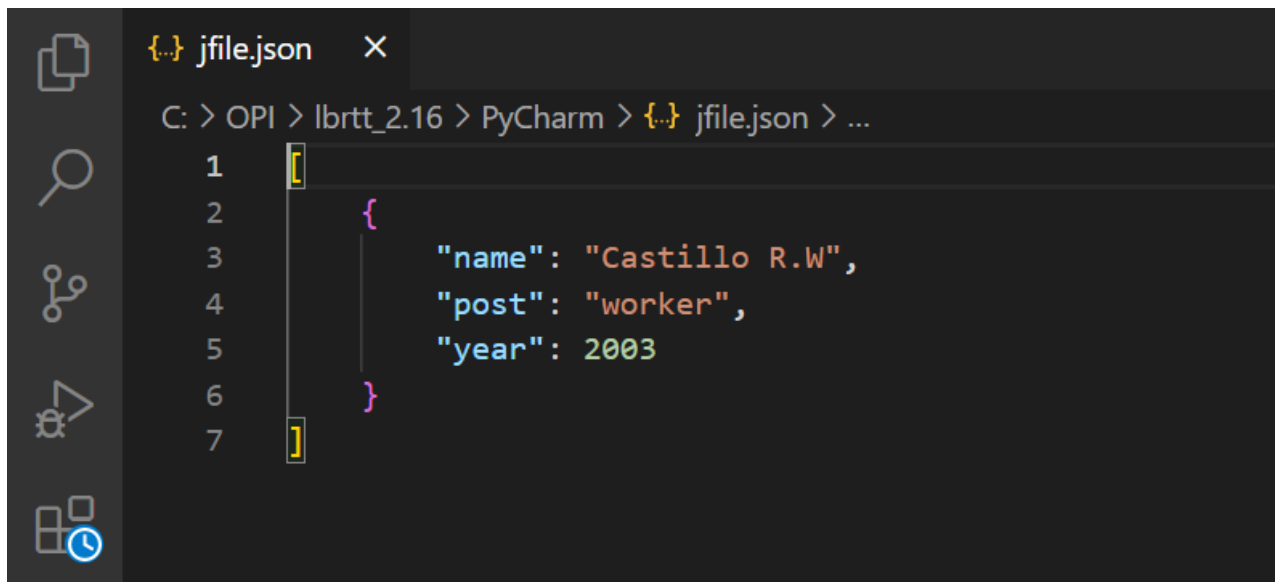


Рисунок 4 – Содержимое файла `jfile.json`

3. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

Код программы:

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

import json
import sys

def get_flight():
    """
    requesting information about the flight
    """
    dst = input("What destination do you need? ")
    nmb = int(input("Which number of the flight do you need? "))
    tpe = input("Which type of plane do you need? ")

    #creation of dictionary
    return {
        'destination': dst,
        'number_flight': nmb,
        'type_plane': tpe,
    }

def display_flights(flights):
    """
    displaying the given information
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 18
        )
```

```

        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                "№",
                "Destination",
                "Number of the flight",
                "Type of the plane"
            )
        )
        print(line)
        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                    idx,
                    flight.get('destination', ''),
                    flight.get('number_flight', ''),
                    flight.get('type_plane', 0)
                )
            )
        print(line)
    else:
        print('list is empty')

def select_flights(flights, t):
    """
    selecting flights that are appropriate
    """
    result = []
    for flight in flights:
        if t in str(flight.values()):
            result.append(flight)
    return result

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

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

def main():
    """
    main function of the program
    """
    #list of the flights
    flights = []

    #organization of an endless loop
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break
        elif command == 'add':
            flight = get_flight()
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(key=lambda item: item.get('destination', ''))
        elif command == 'list':
            display_flights(flights)
        elif command.startswith('select'):

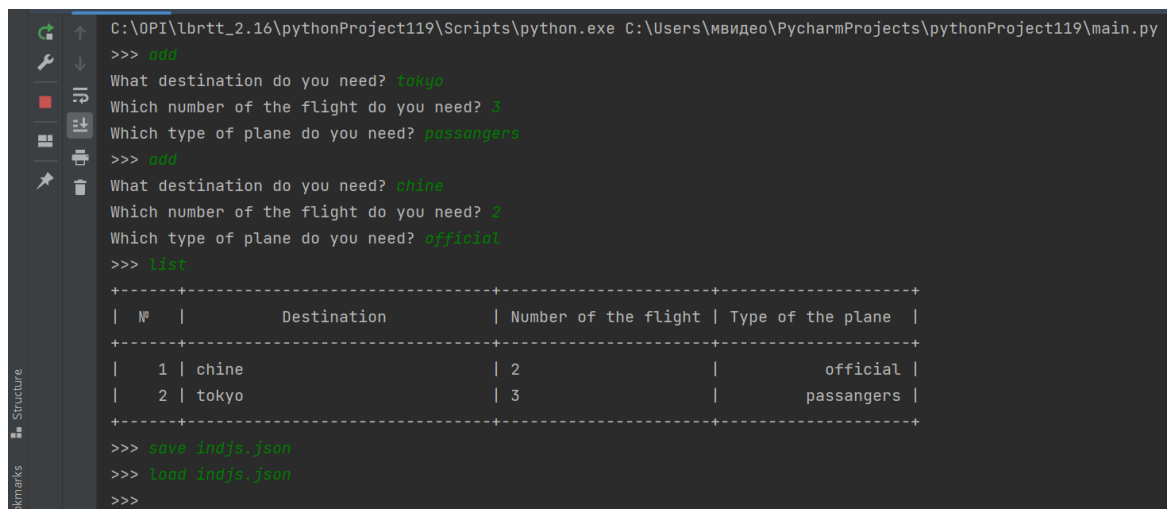
```

```

        a = input("choose type of the plane: ")
        selected = select_flights(flights, a)
        display_flights(selected)
    elif command.startswith("save "):
        parts = command.split(maxsplit=1)
        file_name = parts[1]
        save_flights(file_name, flights)
    elif command.startswith("load "):
        parts = command.split(maxsplit=1)
        file_name = parts[1]
        flights = load_flights(file_name)
    elif command == 'help':
        print("command list:\n")
        print("add - add information about a flight;")
        print("list - display the flight schedule;")
        print("select <type> - select the type of the plane;")
        print("load - download data from file;")
        print("save - save data from file;")
        print("help - show reference;")
        print("exit - leave a program.")
    else:
        print(f"unknown command {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```



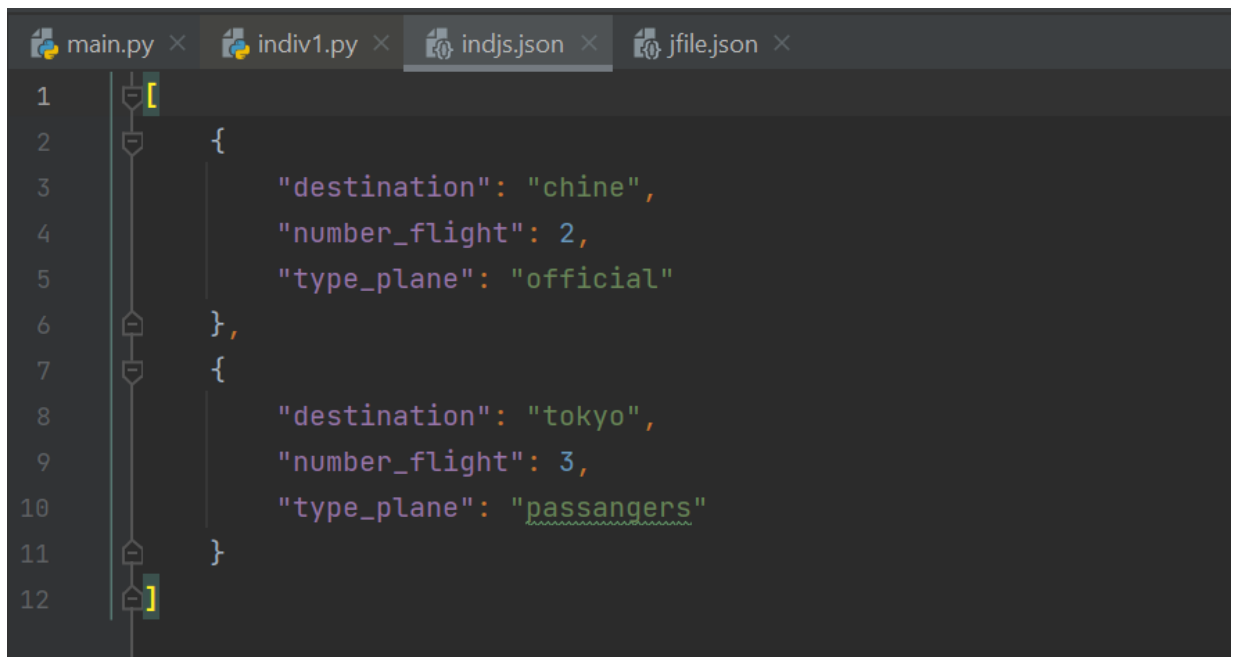
C:\OPI\lbrtt\_2.16\pythonProject119\Scripts\python.exe C:\Users\mvideo\PycharmProjects\pythonProject119\main.py

```

>>> add
What destination do you need? tokyo
Which number of the flight do you need? 3
Which type of plane do you need? passangers
>>> add
What destination do you need? chine
Which number of the flight do you need? 2
Which type of plane do you need? official
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | chine | 2 | official |
| 2 | tokyo | 3 | passangers |
+-----+-----+-----+-----+
>>> save info.json
>>> load info.json
>>>

```

Рисунок 5 – Результат работы программы



```
1  [
2      {
3          "destination": "chine",
4          "number_flight": 2,
5          "type_plane": "official"
6      },
7      {
8          "destination": "tokyo",
9          "number_flight": 3,
10         "type_plane": "passangers"
11     }
12 ]
```

Рисунок 6 – Содержимое файла indjs.json

## Задание повышенной сложности

### Код программы:

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

import json
import sys
import jsonschema

def get_flight()::
    """
    requesting information about the flight
    """
    dst = input("What destination do you need? ")
    nmb = int(input("Which number of the flight do you need? "))
    tpe = input("Which type of plane do you need? ")

    #creation of dictionary
    return {
        'destination': dst,
        'number_flight': nmb,
        'type_plane': tpe,
    }

def display_flights(flights):
    """
    displaying the given information
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 18
```

```

    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
            "№",
            "Destination",
            "Number of the flight",
            "Type of the plane"
        )
    )
    print(line)
    for idx, flight in enumerate(flights, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                idx,
                flight.get('destination', ''),
                flight.get('number_flight', ''),
                flight.get('type_plane', 0)
            )
        )
    print(line)
else:
    print('list is empty')

def select_flights(flights, t):
    """
    selecting flights that are appropriate
    """
    result = []
    for flight in flights:
        if t in str(flight.values()):
            result.append(flight)
    return result

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

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

def load_flights(file_name):
    schema = {
        "type": "array",
        "items": [
            {
                "type": "object",
                "properties": {
                    "destination": {
                        "type": "string"
                    },
                    "number_flight": {
                        "type": "integer"
                    },
                    "type_plane": {
                        "type": "string"
                    }
                }
            },
        ],
        "required": [

```

```

        "destination",
        "number_flight",
        "type_plane"
    ]
},
{
    "type": "object",
    "properties": {
        "destination": {
            "type": "string"
        },
        "number_flight": {
            "type": "integer"
        },
        "type_plane": {
            "type": "string"
        }
    },
    "required": [
        "destination",
        "number_flight",
        "type_plane"
    ]
}
]
}

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("Validation was successful")
    except jsonschema.exceptions.ValidationError:
        print("Validation error", file=sys.stderr)
        exit()
    return loadfile

def main():
    """
    main function of the program
    """
    #list of the flights
    flights = []

    #organization of an endless loop
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break
        elif command == 'add':
            flight = get_flight()
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(key=lambda item: item.get('destination', ''))
        elif command == 'list':
            display_flights(flights)
        elif command.startswith('select'):
            a = input("choose type of the plane: ")
            selected = select_flights(flights, a)
            display_flights(selected)
        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_flights(file_name, flights)

```



```

elif command.startswith("load "):
    parts = command.split(maxsplit=1)
    file_name = parts[1]
    flights = load_flights(file_name)
elif command == 'help':
    print("command list:\n")
    print("add - add information about a flight;")
    print("list - display the flight schedule;")
    print("select <type> - select the type of the plane;")
    print("load - download data from file;")
    print("save - save data from file;")
    print("help - show reference;")
    print("exit - leave a program.")
else:
    print(f"unknown command {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

```

Run: main x
C:\OPI\lbrtt_2.16\pythonProject120\Scripts\python.exe C:\Users\мвидео\PycharmPr
>>> add
What destination do you need? tokyo
Which number of the flight do you need? 3
Which type of plane do you need? official
>>> add
What destination do you need? china
Which number of the flight do you need? 2
Which type of plane do you need? passangers
>>> save ind_js2.json
>>> load ind_js2.json
Validation was successful
>>>

```

Рисунок 7 – Результат работы программы

Контр. вопросы и ответы на них:

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как JAYsən)

– текстовый формат обмена данными, основанный на JavaScript. Как многие

другие текстовые форматы, JSON легко читается людьми.

2. Какие типы значений используются в JSON?

Набор пар ключ: значение. Упорядоченный набор значений.

### 3. Как организована работа со сложными данными в JSON?

JSON может содержать другие вложенные объекты в JSON, в дополнение к вложенным массивам. Такие объекты и массивы будут передаваться, как значения назначенные ключам и будут представлять собой связку ключ-значение.

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

Формат обмена данными JSON5 (JSON5) - это надмножество JSON, целью которого является смягчение некоторых ограничений JSON путем расширения его синтаксиса для включения некоторых продуктов из ECMAScript 5.1. Эта библиотека JavaScript является официальной эталонной реализацией библиотек синтаксического анализа и сериализации JSON5. Краткое описание возможностей. Следующие функции ECMAScript 5.1, которые не поддерживаются в JSON, были расширены до JSON5. Объекты. Ключи объекта могут быть идентификатором ECMAScript 5.1.

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

Реализация Python формата данных JSON5. JSON5 расширяет формат обмена данными JSON, чтобы сделать его более удобным для использования в качестве языка конфигурации:

Комментарии в стиле JavaScript (как однострочные, так и многострочные) разрешены.

Ключи объектов могут быть без кавычек, если они являются допустимыми идентификаторами ECMAScript.

Объекты и массивы могут заканчиваться запятыми.

Строки могут заключаться в одинарные кавычки, допускаются

многострочные строковые литералы.

Есть еще несколько более мелких расширений JSON; см. полную информацию на странице выше.

Этот проект реализует реализацию чтения и записи для Python; где возможно, он отражает стандартный пакет Python JSON API для простоты использования.

Есть одно заметное отличие от JSON api: методы `load ()` и `load ()` поддерживают опциональную проверку (и отклонение) повторяющихся ключей объекта; `pass allow_duplicate_keys = False` для этого (по умолчанию разрешены дубликаты).

Это ранний выпуск. Это было достаточно хорошо протестировано, но это МЕДЛЕННО. Он может быть в 1000-6000 раз медленнее, чем модуль JSON, оптимизированный для C, и в 200 раз (или более) медленнее, чем модуль JSON на чистом Python.

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

```
json.dump() # конвертировать python объект в json и записать в файл  
json.dumps() # тоже самое, но в строку.
```

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

Dumps записывает в строку, а dump в файл.

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

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

```
import codecs  
json.load(codecs.open('sample.json', 'r', 'utf-8-sig'))
```

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных?

Схема JSON - это словарь, который позволяет аннотировать и проверять документы JSON.

Преимущества:

- Описывает ваш существующий формат (ы) данных.
- Предоставляет понятную документацию, читаемую человеком и машиной.
- Проверяет данные, которые полезны для:
- Автоматизированное тестирование.
- Обеспечение качества предоставленных клиентом данных.