МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.16 по дисциплине основы программнойинженерии

Выполнил:
Кожухов Филипп Денисович,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной
безопасности,Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

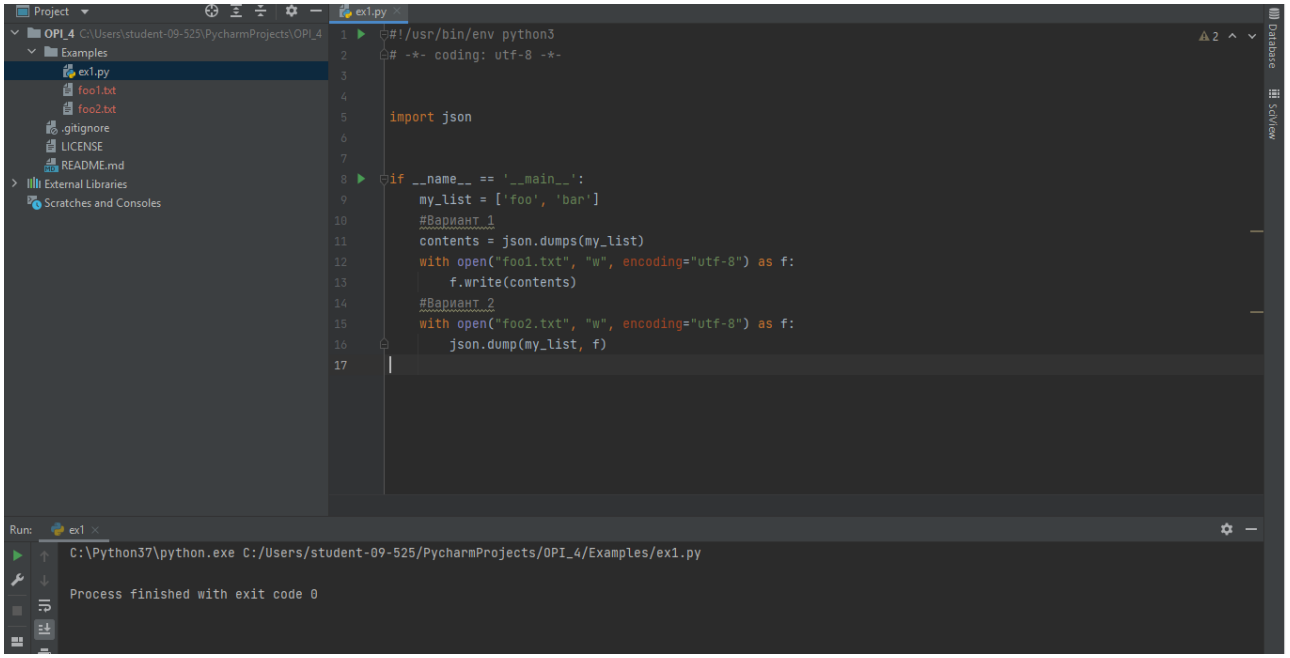


Рисунок 1 - Пример №1

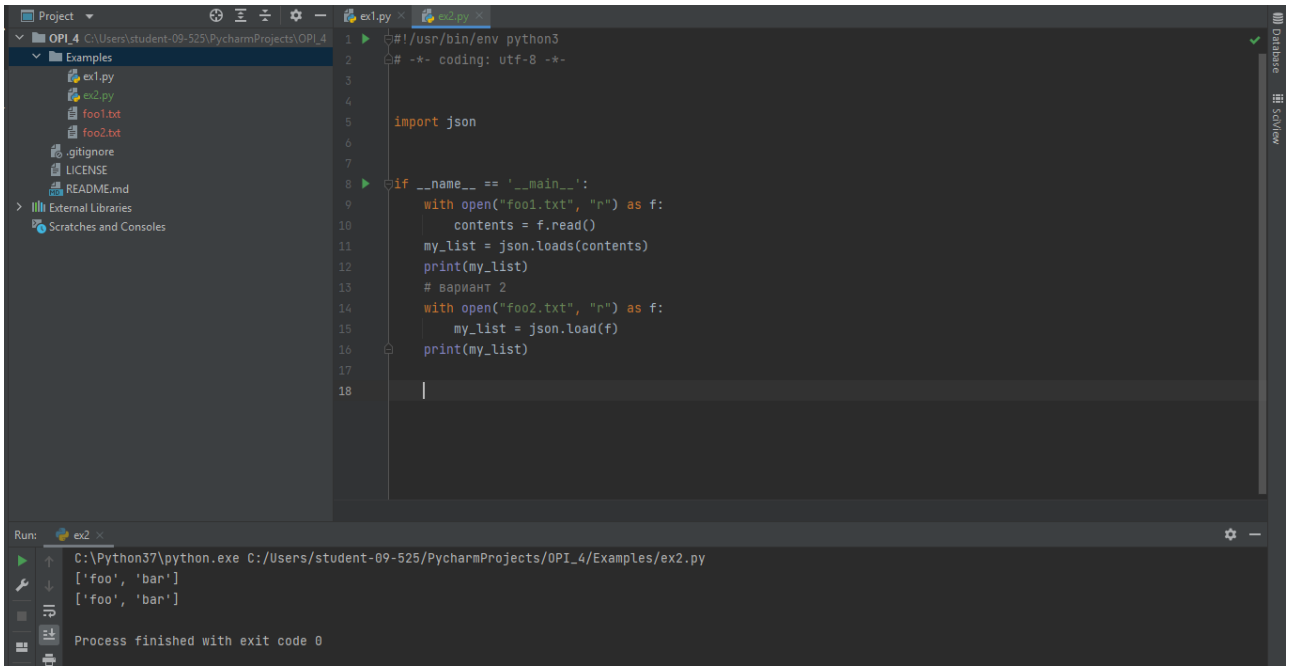
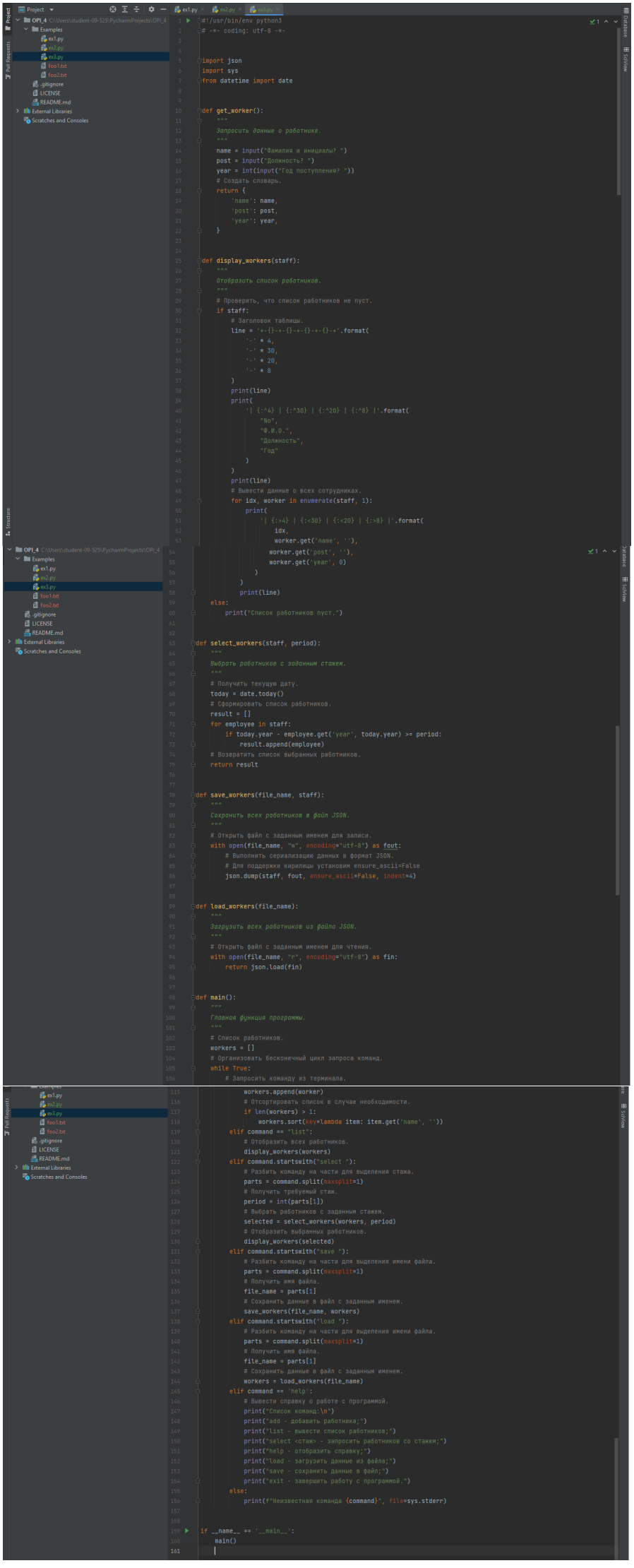


Рисунок 2 - Пример №2



```
#!/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 кирилицы 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':
            # Выход из программы.
            break
        elif command == 'add':
            # Запросить данные о работнике.
            worker = get_worker()
            # Добавить нового работника.
            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 3 - Пример №3

```
C:\Python37\python.exe C:/Users/student-09-525/PycharmProjects/OPI_4/Examples/ex3.py
>>> load backup.txt
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | JSdoijq K.O                    | Shdqwnm              |     2014 |
+------+--------------------------------+----------------------+----------+
|    2 | Kodsodq D.F                    | Jksdjqw              |     1994 |
+------+--------------------------------+----------------------+----------+
|    3 | Sdklakfds L.V                  | KJdzsj               |     2004 |
+------+--------------------------------+----------------------+----------+
>>> exit

Process finished with exit code 0
```

Рисунок 4 - Файл backup.txt из примера №3

```
[
    {
        "name": "JSdoijq K.O",
        "post": "Shdqwnm",
        "year": 2014
    },
    {
        "name": "Kodsodq D.F",
        "post": "Jksdjqw",
        "year": 1994
    },
    {
        "name": "Sdklakfds L.V",
        "post": "KJdzsj",
        "year": 2004
    }
]
```

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

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


import json
import jsonschema
from jsonschema import validate
import sys



def get_flight():
    """
    Запросить данные о полёте
    """
    flight_destination = input("Введите название пункта назначения ")
    flight_number = input("Введите номер рейса ")
    airplane_type = input("Введите тип самолета ")
    return {
        'flight_destination': flight_destination,
        'flight_number': flight_number,
        'airplane_type': airplane_type,
    }


def display_flights(flights):
    """
     Отобразить список рейсов
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Пункт назначения",
                "Номер рейса",
                "Тип самолета"
            )
        )
        print(line)

        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
                    idx,
                    flight.get('flight_destination', ''),
                    flight.get('flight_number', ''),
                    flight.get('airplane_type', 0)
                )
            )
        print(line)

    else:
        print("Список рейсов пуст")


def select_flights(flights, airplane_type):
    """
    Выбрать рейсы самолётов заданного типа
    """
    count = 0
    res = []
    for flight in flights:
        if flight.get('airplane_type') == airplane_type:
            count += 1
            res.append(flight)
    if count == 0:
        print("рейсы не найдены")

    return res


def save_workers(file_name, planes):
    """
    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 кирилицы установим ensure_ascii=False
        json.dump(planes, 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)
```

```python
def main():
    """
    Главная функция программы
    """
    with open('schema_test.json', 'r') as file:
        schema = json.load(file)
    flights = []
    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('flight_destination', ''))

        elif command == 'list':
            display_flights(flights)

        elif command.startswith('select '):
            parts = command.split(' ', maxsplit=1)
            airplane_type = (parts[1].capitalize())
            print(f"Для типа самолета {airplane_type}:")
            selected = select_flights(flights, airplane_type)
            display_flights(selected)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить рейс;")
            print("list - вывести список всех рейсов;")
            print("select <тип самолета> - запросить рейсы указанного типа "
                  "самолета;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        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, flights)


        elif command.startswith("load "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            flights = load_workers(file_name)
            try:
                validate(instance=flights, schema=schema)
            except jsonschema.exceptions.ValidationError as err:
                err = "Given JSON data are InValid"
                print(err)
            message = "Given JSON data are Valid"
            print(message)
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


if __name__ == '__main__':
    main()
```

Рисунки 5-8 - Код ИДЗ

Рисунок 9 - Вывод программы

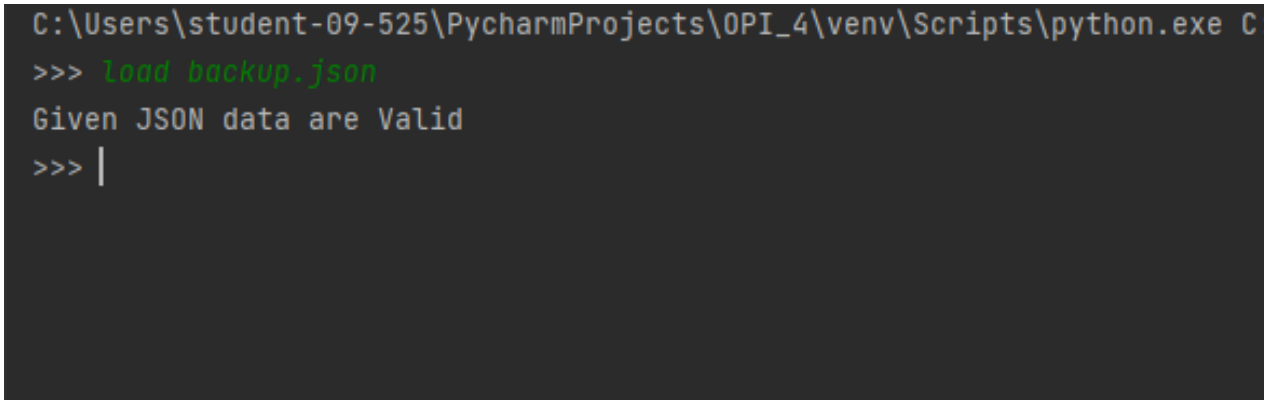
```
C:\Python37\python.exe C:/Users/student-09-525/PycharmProjects/OPI_4/Examples/ex3.py
>>> help
Список команд:

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.
>>> add
Фамилия и инициалы? Sdjkajvccsq K.K
Должность? Sdjkjkcq
Год поступления? 2004
>>> add
Фамилия и инициалы? Wcodos H.F
Должность? Andkj2q
Год поступления? 1998
>>> add
Фамилия и инициалы? Hqndskjx L.Q
Должность? Nciuqnsm
Год поступления? 2021
>>> add
Фамилия и инициалы? Kojhq U.L
Должность? Unqciuqskld
Год поступления? 2016
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Hqndskjx L.Q                   | Nciuqnsm             |     2021 |
+------+--------------------------------+----------------------+----------+
|    2 | Kojhq U.L                      | Unqciuqskld          |     2016 |
+------+--------------------------------+----------------------+----------+
|    3 | Sdjkajvccsq K.K                | Sdjkjkcq             |     2004 |
+------+--------------------------------+----------------------+----------+
|    4 | Wcodos H.F                     | Andkj2q              |     1998 |
+------+--------------------------------+----------------------+----------+
>>> save backup.json
>>> exit
```

Рисунок 10 - Содержимое файла backup.json

Рисунок 11 - Валидация


```
[
    {
        "name": "Hqndskjx L.Q",
        "post": "Nciuqnsm",
        "year": 2021
    },
    {
        "name": "Kojhq U.L",
        "post": "Unqciuqskld",
        "year": 2016
    },
    {
        "name": "Sdjkajvccsq K.K",
        "post": "Sdjkjkcq",
        "year": 2004
    },
    {
        "name": "Wcodos H.F",
        "post": "Andkj2q ",
        "year": 1998
    }
]
```



ОТВЕТЫ НА ВОПРОСЫ

1. Для чего используется JSON?

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента.

Объект JSON это формат данных — ключ-значение, который обычно рендерится в фигурных скобках. Когда вы работаете с JSON, то вы скорее

всего видите JSON объекты в .json файле, но они также могут быть и как JSON объект или строка уже в контексте самой программы.

2. Какие типы значений используются в JSON?

Если быть точным, то им нужно быть одним из шести типов данных: строкой, числом, объектом, массивом, булевым значением или null.

Как было показано ранее JSON-текст представляет собой (в закодированном виде) одну из двух структур:

Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимость не регулируется стандартом, это остаётся на усмотрение программного обеспечения. Как правило, регистр учитывается программами — имена с буквами в разных регистрах считаются разными, значением — любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.

Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

В качестве значений в JSON могут быть использованы:

запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.

массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.

число (целое или вещественное).

литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.

Строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape- последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты ', ", \, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

3. Как организована работа со сложными данными в JSON?

Вложенные объекты

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. Фигурные скобки везде используются для формирования вложенного JSON объекта с ассоциированными именами пользователей и данными локаций для каждого из них. Как и с любым другим значением, используя объекты, двоеточие используется для разделения элементов.

```
{
"sammy" : {
"username" : "SammyShark", "location" : "Indian Ocean", "online" : true,
"followers" : 987
},
"jesse" : {
"username" : "JesseOctopus", "location" : "Pacific Ocean", "online" : false,
"followers" : 432
},
"drew" : {
"username" : "DrewSquid", "location" : "Atlantic Ocean", "online" : false,
"followers" : 321
},

"jamie" : {
"username" : "JamieMantisShrimp", "location" : "Pacific Ocean", "online" : true,
"followers" : 654
}
}
```

Вложенные массивы

Данные также могут быть вложены в формате JSON, используя JavaScript массивы, которые передаются как значения. JavaScript использует квадратные скобки [ ] для формирования массива. Массивы по своей сути — это упорядоченные коллекции и могут включать в себя значения совершенно разных типов данных. Мы можем использовать массив при работе с большим количеством данных, которые могут быть легко сгруппированны вместе, как например, если есть несколько разных сайтов и профайлов в социальных сетях ассоциированных с одним пользователем.

```
{
```

```
"first_name" : "Sammy",
"last_name" : "Shark",
"location" : "Ocean",
"websites" : [
{
"description" : "work",
"URL" : "https://www.digitalocean.com/"
},
{
"desciption" : "tutorials",
"URL" : "https://www.digitalocean.com/community/tutorials" }
],
"social_media" : [

{
"description" : "twitter",
"link" : "https://twitter.com/digitalocean"
},
{
"description" : "facebook",
"link" : "https://www.facebook.com/DigitalOceanCloudHosting" },
{
"description" : "github",
"link" : "https://github.com/digitalocean"
}
]
}
```

Ключи "websites" и "social_media" используют массив для вложения информации о сайтах пользователя и профайлов в социальных сетях. Мы знаем, что это массивы — из-за квадратных скобок.

Использование вложенности в нашем JSON формате позволяет нам работать с наиболее сложными и иерархичными данными.

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. Для некоторых языков программирования уже существуют

парсеры json5.

Некоторые нововведения:

* Поддерживаются как однострочные //, так и многострочные /* */ комментарии.

* Записи и списки могут иметь запятую после последнего элемента (удобно при копировании элементов).

* Ключи записей могут быть без кавычек, если они являются валидными идентификаторами ECMAScript 5.

* Строки могут заключаться как в одинарные, так и в двойные кавычки.

* Числа могут быть в шестнадцатеричном виде, начинаться или заканчиваться десятичной точкой, включать Infinity, -Infinity, NaN и -NaN, начинаться со знака +.

Проще говоря, он убирает некоторые ограничения JSON, расширяя его синтаксис.

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

Существует пакет PyJSON5, который содержит множество функций для расширения функционала JSON.

Ниже представлены функции для сериализации данных

Функции для кодирования/декодирования данных:

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

Сериализация данных в формат JSON:

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

Обе эти функции принимают следующие необязательные аргументы:

Если skipkeys = True , то ключи словаря не базового типа (str , int , float , bool , None) будут проигнорированы, вместо того, чтобы вызывать исключение TypeError .

Если ensure_ascii = True , все не-ASCII символы в выводе будут экранированы последовательностями \uXXXX , и результатом будет строка, содержащая только ASCII символы. Если ensure_ascii = False , строки запишутся как есть.

Если check_circular = False , то проверка циклических ссылок будет пропущена, а такие ссылки будут вызывать OverflowError .

Если allow_nan = False , при попытке сериализовать значение с запятой, выходящее за допустимые пределы, будет вызываться ValueError (nan, inf, -

inf) в строгом соответствии со спецификацией JSON, вместо того, чтобы использовать эквиваленты из JavaScript (NaN, Infinity, -Infinity).

Если indent является неотрицательным числом, то массивы и объекты в JSON будут выводиться с этим уровнем отступа. Если уровень отступа 0, отрицательный или "", то вместо этого будут просто использоваться новые строки. Значение по умолчанию None отражает наиболее компактное представление. Если indent - строка, то она и будет использоваться в качестве отступа.

Если sort_keys = True , то ключи выводимого словаря будут отсортированы.

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

json.dumps() конвертирует python объект в json и записывает его в строку вместо записи в файл.

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

Десериализация 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)

Обе эти функции принимают следующие аргументы:

object_hook - опциональная функция, которая применяется к результату декодирования объекта (dict). Использоваться будет значение, возвращаемое этой функцией, а не полученный словарь.

object_pairs_hook - опциональная функция, которая применяется к результату декодирования объекта с определённой последовательностью пар ключ/ значение. Будет использован результат, возвращаемый функцией, вместо исходного словаря. Если задан так же object_hook , то приоритет отдаётся object_pairs_hook .

parse_float , если определён, будет вызван для каждого значения JSON с плавающей точкой. По умолчанию, это эквивалентно float(num_str) .

parse_int , если определён, будет вызван для строки JSON с числовым значением. По умолчанию эквивалентно int(num_str) .

parse_constant , если определён, будет вызван для следующих строк: "- Infinity", "Infinity", "NaN". Может быть использовано для возбуждения исключений при обнаружении ошибочных чисел JSON.

Если не удастся десериализовать JSON, будет возбуждено исключение ValueError .

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

Использование кодировки UTF-8 или ensure_ascii=False

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