

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

**Отчет по лабораторной работе № 4
«Работа с данными формата JSON в языке Python»**

по дисциплине «Основы программной инженерии»

Выполнила:
Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

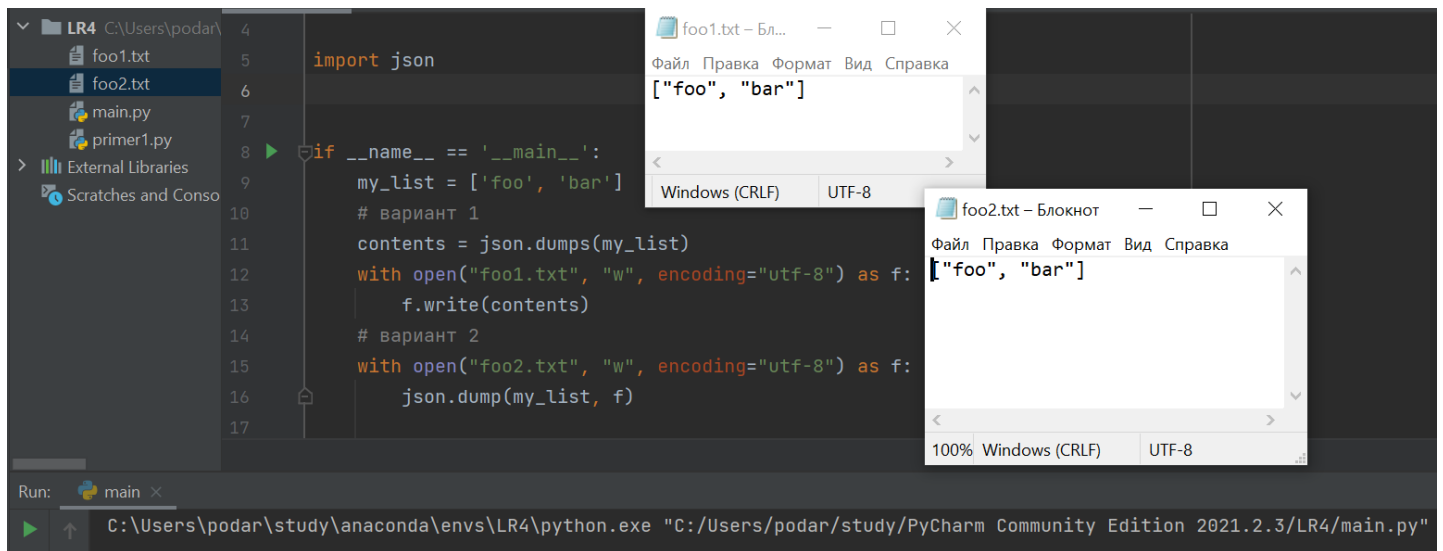


Рисунок 1 - Запись списка или словаря в файл

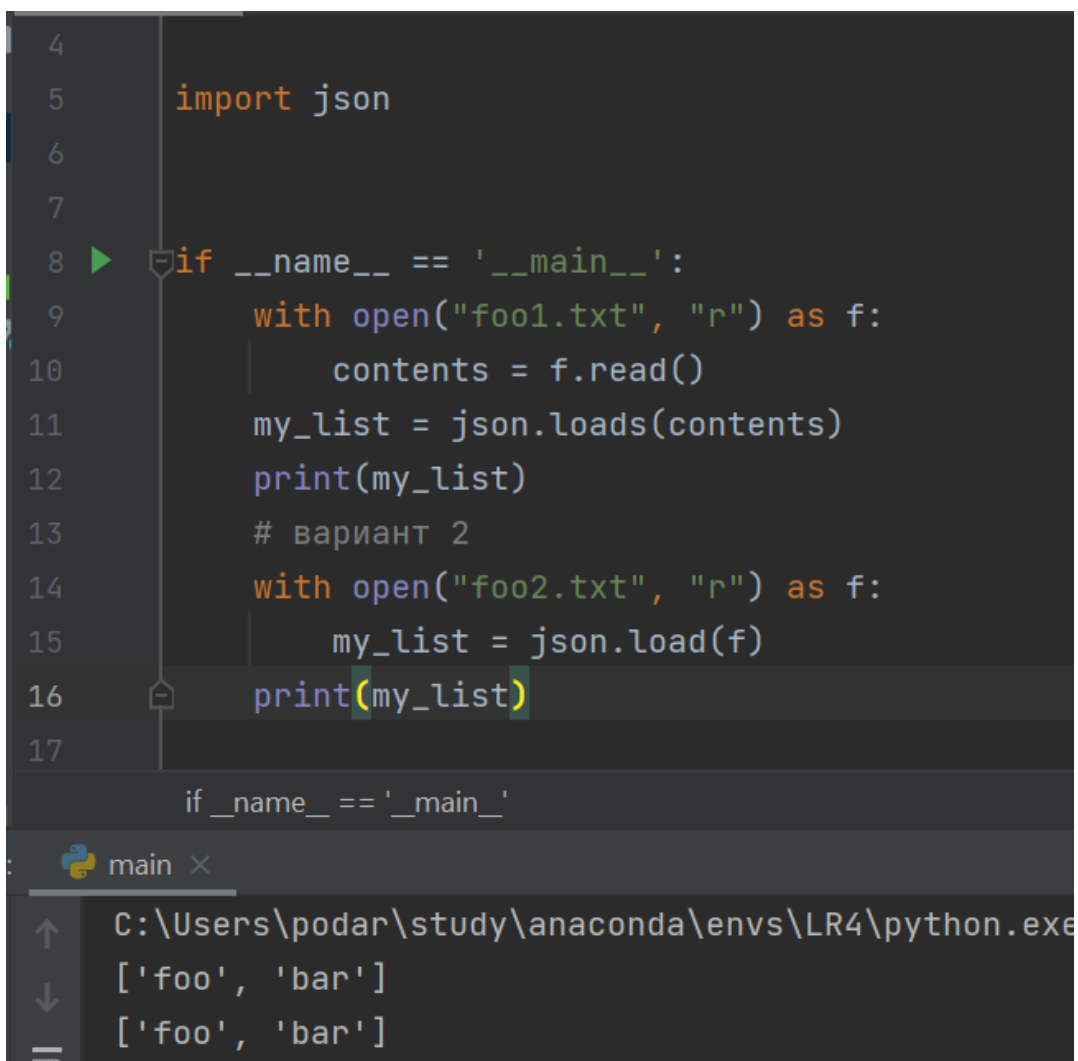


Рисунок 2 - Чтение списка или словаря из файла

```
# 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):
```

Рисунок 3 – Код примера

```

C:\Users\podar\study\anaconda\envs\LR4\
>>> add
Фамилия и инициалы? Первых Д.А.
Должность? студент
Год поступления? 2020
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? студент
Год поступления? 2019
>>> add
Фамилия и инициалы? Петров П.П.
Должность? студент
Год поступления? 2021

```

Рисунок 4 – Результат выполнения команды add

```

>>> list
+-----+-----+-----+-----+
| No  |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|   1  | Иванов И.И.              | студент              |  2019   |
+-----+-----+-----+-----+
|   2  | Первых Д.А.              | студент              |  2020   |
+-----+-----+-----+-----+
|   3  | Петров П.П.              | студент              |  2021   |
+-----+-----+-----+-----+

```

Рисунок 5 – Результат выполнения команды list

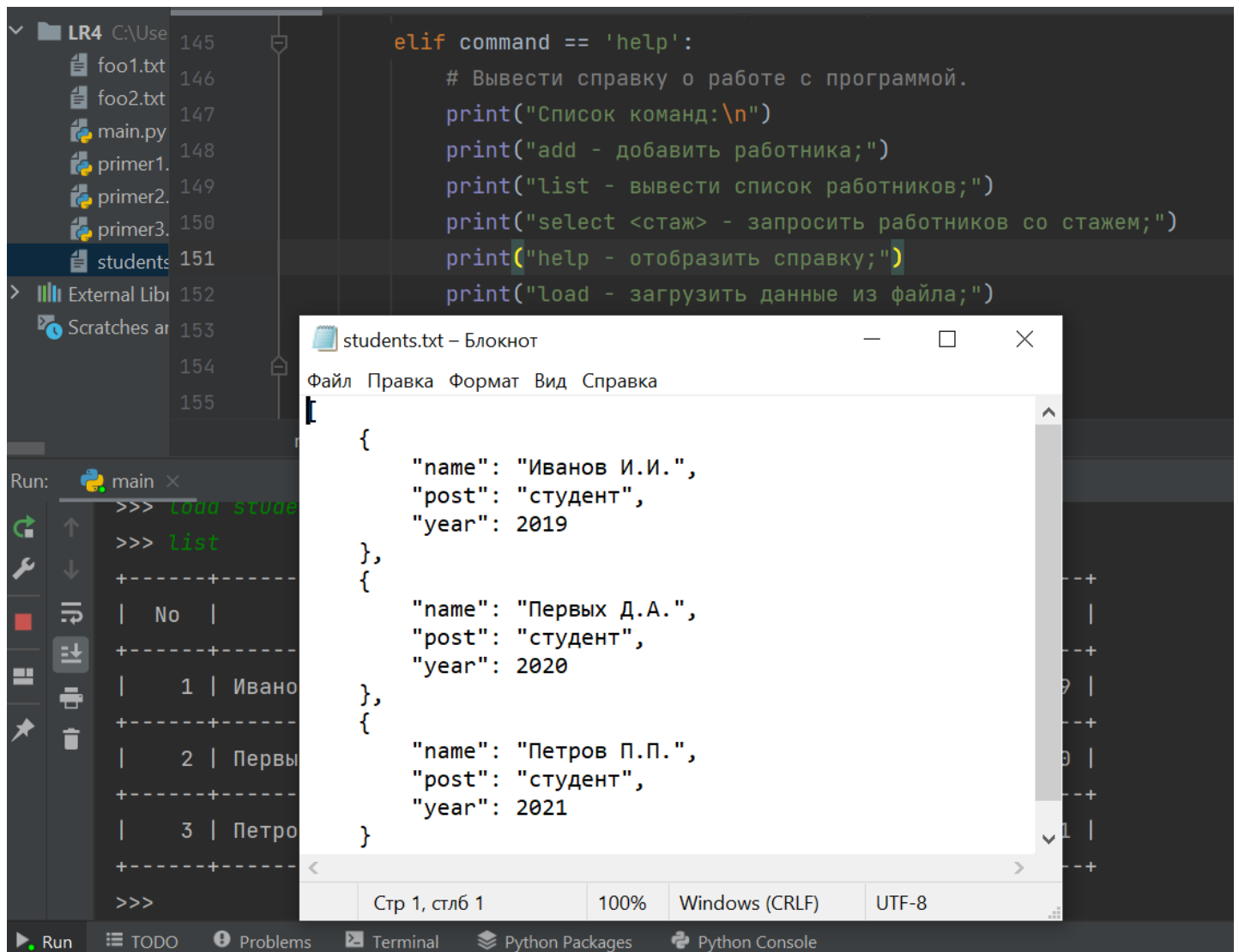


Рисунок 6 – Результат работы программы

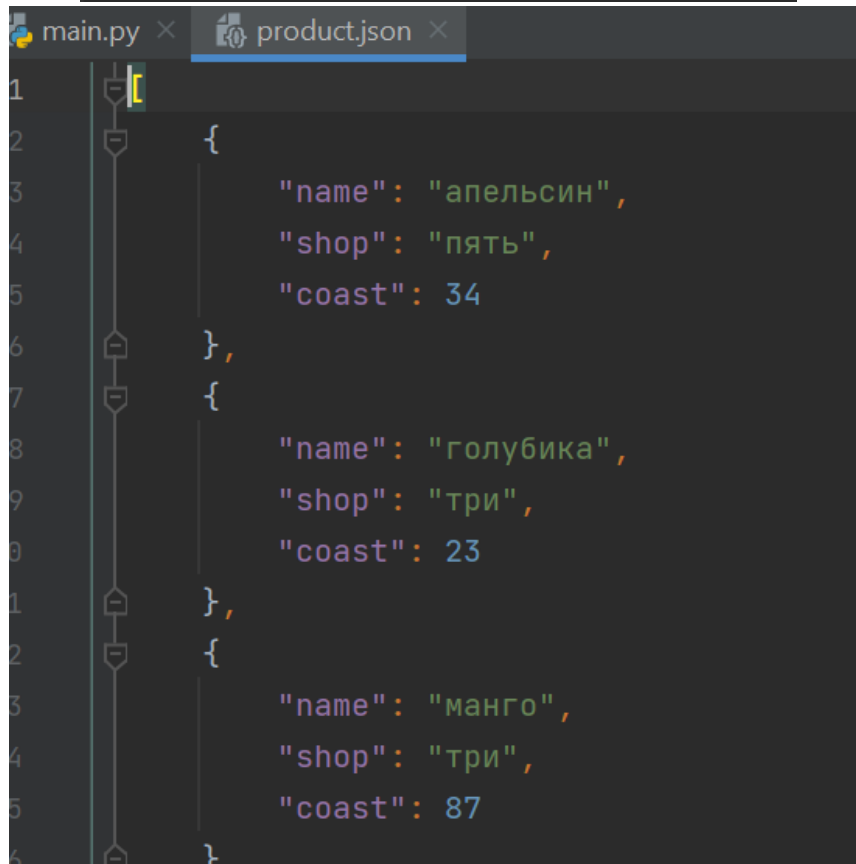
Индивидуальное задание.

```
>>> add
Название товара? голубика
Название магазина? три
Введите его цену 43
>>> add
Название товара? апельси
Название магазина? пять
Введите его цену 45
>>> add
Название товара? манго
Название магазина? три
Введите его цену 43
>>> list
```

| № | Наименование товара | Название магазина | Стоимость |
|---|---------------------|-------------------|-----------|
| 1 | апельси | пять | 45 |
| 2 | голубика | три | 43 |
| 3 | манго | три | 43 |

Рисунок 7 – Результат работы программы

```
Список продуктов пуст
>>> save product.json
>>> load product.json
```



```
{
  "name": "апельсин",
  "shop": "пять",
  "coast": 34
},
{
  "name": "голубика",
  "shop": "три",
  "coast": 23
},
{
  "name": "манго",
  "shop": "три",
  "coast": 87
}
```

Рисунок 8 – Результат работы программы

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 чертой «\» (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 сгруппированы 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/"  
    },  
    {
```

```
"description" : "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

Quick Encoder Summary

| | |
|--------------------------------------------------------------|-----------------------------------------------------------------------|
| <code>encode (data, *[, options])</code> | Serializes a Python object as a JSON5 compatible string. |
| <code>encode_bytes (data, *[, options])</code> | Serializes a Python object to a JSON5 compatible bytes string. |
| <code>encode_callback (data, cb[, supply_bytes, ...])</code> | Serializes a Python object into a callback function. |
| <code>encode_io (data, fp[, supply_bytes, options])</code> | Serializes a Python object into a file-object. |
| <code>encode_noop (data, *[, options])</code> | Test if the input is serializable. |
| <code>dump (obj, fp, **kw)</code> | Serializes a Python object to a JSON5 compatible string. |
| <code>dumps (obj, **kw)</code> | Serializes a Python object to a JSON5 compatible string. |
| <code>Options</code> | Customizations for the <code>encoder_*(...)</code> function family. |
| <code>Json5EncoderException</code> | Base class of any exception thrown by the serializer. |
| <code>Json5UnstringifiableType ((message, ...))</code> | The encoder was not able to stringify the input, or it was too large. |

Функции для кодирования/декодирования данных:

Quick Decoder Summary

| | |
|----------------------------------------------------------------|-----------------------------------------------------------------|
| <code>decode (data[, maxdepth, some])</code> | Decodes JSON5 serialized data from an <code>str</code> object. |
| <code>decode_latin1 (data[, maxdepth, some])</code> | Decodes JSON5 serialized data from a <code>bytes</code> object. |
| <code>decode_buffer (obj[, maxdepth, some, wordlength])</code> | Decodes JSON5 serialized data from an object that s |
| <code>decode_callback (cb[, maxdepth, some, args])</code> | Decodes JSON5 serialized data by invoking a callback |
| <code>decode_io (fp[, maxdepth, some])</code> | Decodes JSON5 serialized data from a file-like object |
| <code>load (fp, **kw)</code> | Decodes JSON5 serialized data from a file-like object |
| <code>loads (s, *[, encoding])</code> | Decodes JSON5 serialized data from a string. |
| <code>Json5DecoderException ([message, result])</code> | Base class of any exception thrown by the parser. |
| <code>Json5NestingTooDeep</code> | The maximum nesting level on the input data was ex |
| <code>Json5EOF</code> | The input ended prematurely. |
| <code>Json5IllegalCharacter ([message, result, ...])</code> | An unexpected character was encountered. |
| <code>Json5ExtraData ([message, result, character])</code> | The input contained extraneous data. |
| <code>Json5IllegalType ([message, result, value])</code> | The user supplied callback function returned illegal d |

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). Используется 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`.

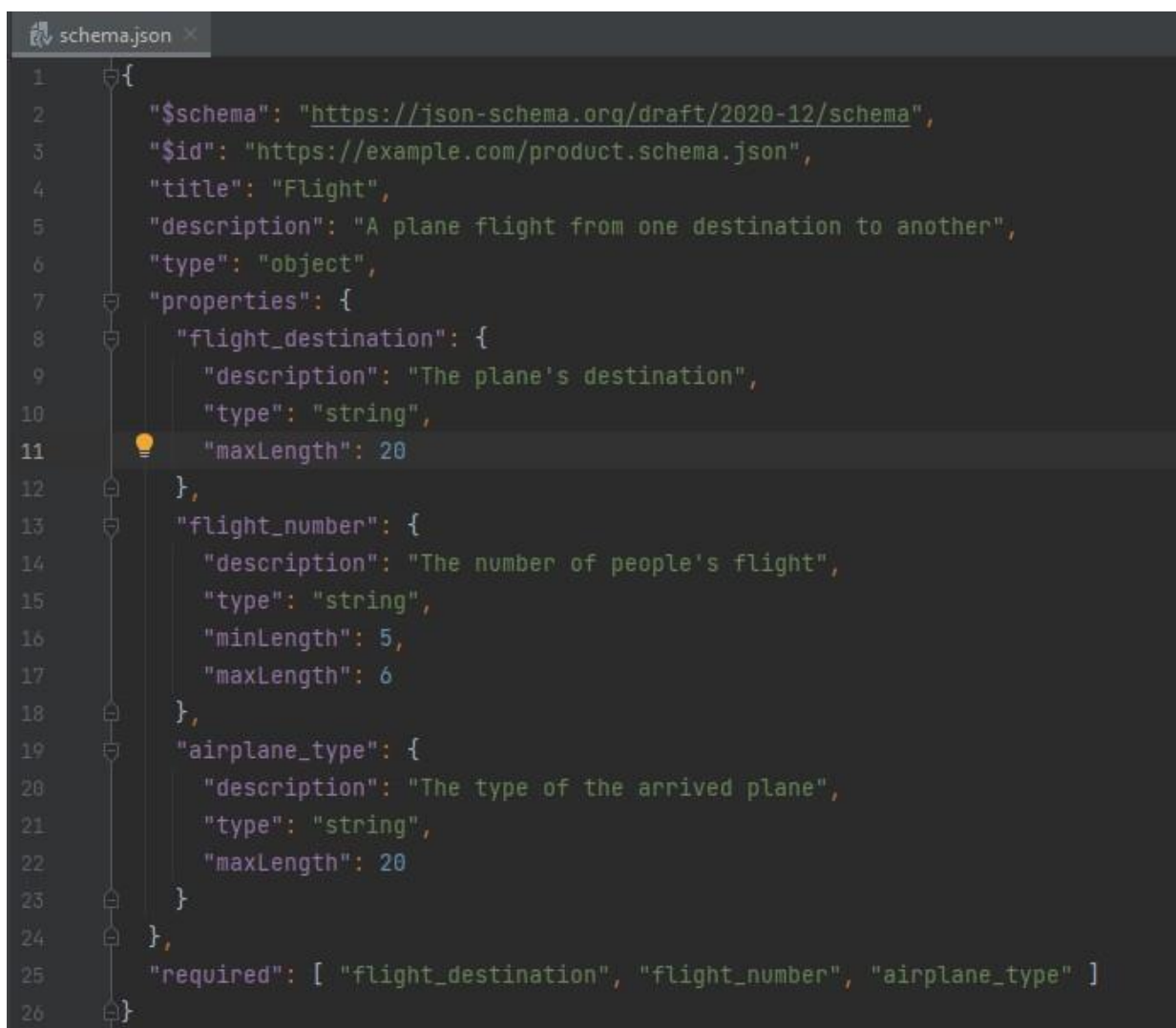
9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кириллицу?

Использование кодировки UTF-8 или `ensure_ascii=False`

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

Приведите схему данных для примера 1.

Схема данных представляет собой код, который используется для валидации данных в формате JSON. Схема данных:



```
1  {
2    "$schema": "https://json-schema.org/draft/2020-12/schema",
3    "$id": "https://example.com/product.schema.json",
4    "title": "Flight",
5    "description": "A plane flight from one destination to another",
6    "type": "object",
7    "properties": {
8      "flight_destination": {
9        "description": "The plane's destination",
10       "type": "string",
11       "maxLength": 20
12     },
13     "flight_number": {
14       "description": "The number of people's flight",
15       "type": "string",
16       "minLength": 5,
17       "maxLength": 6
18     },
19     "airplane_type": {
20       "description": "The type of the arrived plane",
21       "type": "string",
22       "maxLength": 20
23     }
24   },
25   "required": [ "flight_destination", "flight_number", "airplane_type" ]
26 }
```