

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 №4 по дисциплине  
«Основы программной инженерии»

Выполнил студент  
2 курса, группы ПИЖ-б-о-20-1  
Тотубалина С.С.

Проверил:  
Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

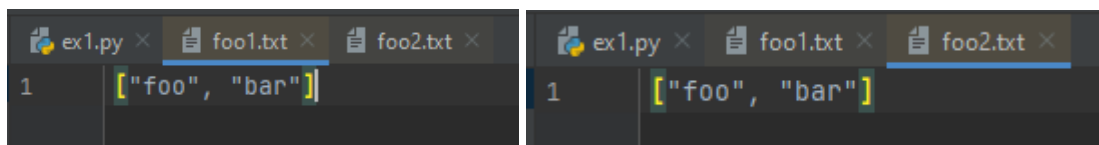
Ставрополь, 2022 г

## Ход работы

### 1. Проработка примеров лабораторной работы.

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 import json
6
7
8 ▶ if __name__ == '__main__':
9     my_list = ['foo', 'bar']
10    # вариант 1
11    contents = json.dumps(my_list)
12    with open("foo1.txt", "w", encoding="utf-8") as f:
13        f.write(contents)
14    # вариант 2
15    with open("foo2.txt", "w", encoding="utf-8") as f:
16        json.dump(my_list, f)
```

Рисунок 1.1 – код программы ex1.py



The image shows two side-by-side screenshots of a text editor. The left window shows 'foo1.txt' with the content `[\"foo\", \"bar\"]`. The right window shows 'foo2.txt' with the same content `[\"foo\", \"bar\"]`. Both windows have tabs for 'ex1.py', 'foo1.txt', and 'foo2.txt' at the top.

Рисунок 1.2 – результат работы программы ex1.py

```
ex2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 import json
6
7
8 ▶ if __name__ == '__main__':
9     with open("foo1.txt", "r") as f:
10         contents = f.read()
11         my_list = json.loads(contents)
12         print(my_list)
13     # вариант 2
14     with open("foo2.txt", "r") as f:
15         my_list = json.load(f)
16     print(my_list)
```

Рисунок 1.3 – код программы ex2.py

```
C:\Users\admin\anaconda3\python.exe C:/Users/admin/lab.2.16/example/ex2.py
['foo', 'bar']
['foo', 'bar']

Process finished with exit code 0
```

Рисунок 1.4 – результат работы программы ex2.py

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import json
6      import sys
7      from datetime import date
8
9
10     def get_worker():
11         """
12         Запросить данные о работнике.
13         """
14         name = input("Фамилия и инициалы? ")
15         post = input("Должность? ")
16         year = int(input("Год поступления? "))
17         # Создать словарь.
18         return {
19             'name': name,
20             'post': post,
21             'year': year,
22         }
23
24
25     def display_workers(staff):
26         """
27         Отобразить список работников.
28         """
29         # Проверить, что список работников не пуст.
30         if staff:
31             # Заголовок таблицы.
32             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
33                 '-' * 4,
34                 '-' * 30,
35                 '-' * 20,
36                 '-' * 8
37             )
38             print(line)
39             print(
40                 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
41                     "No",
```

Рисунок 1.5 – код программы ex3.py

```

83     with open(file_name, "w", encoding="utf-8") as fout:
84         # Выполнить сериализацию данных в формат JSON.
85         # Для поддержки кириллицы установим ensure_ascii=False
86         json.dump(staff, fout, ensure_ascii=False, indent=4)
87
88
89     def load_workers(file_name):
90         """
91         Загрузить всех работников из файла JSON.
92         """
93         # Открыть файл с заданным именем для чтения.
94         with open(file_name, "r", encoding="utf-8") as fin:
95             return json.load(fin)
96
97
98     def main():
99         """
100         Главная функция программы.
101         """
102         # Список работников.
103         workers = []
104         # Организовать бесконечный цикл запроса команд.
105         while True:
106             # Запросить команду из терминала.
107             command = input(">>> ").lower()
108             # Выполнить действие в соответствие с командой.
109             if command == "exit":
110                 break
111             elif command == "add":
112                 # Запросить данные о работнике.
113                 worker = get_worker()
114                 # Добавить словарь в список.
115                 workers.append(worker)
116                 # Отсортировать список в случае необходимости.
117                 if len(workers) > 1:
118                     workers.sort(key=lambda item: item.get('name', ''))
119             elif command == "list":
120                 # Отобразить всех работников.
121                 display_workers(workers)
122             elif command.startswith("select "):
123                 # Разбить команду на части для выделения стажа.

```

Рисунок 1.6 – код программы ex3.py

```

124     parts = command.split(maxsplit=1)
125     # Получить требуемый стаж.
126     period = int(parts[1])
127     # Выбрать работников с заданным стажем.
128     selected = select_workers(workers, period)
129     # Отобразить выбранных работников.
130     display_workers(selected)
131 elif command.startswith("save "):
132     # Разбить команду на части для выделения имени файла.
133     parts = command.split(maxsplit=1)
134     # Получить имя файла.
135     file_name = parts[1]
136     # Сохранить данные в файл с заданным именем.
137     save_workers(file_name, workers)
138 elif command.startswith("load "):
139     # Разбить команду на части для выделения имени файла.
140     parts = command.split(maxsplit=1)
141     # Получить имя файла.
142     file_name = parts[1]
143     # Сохранить данные в файл с заданным именем.
144     workers = load_workers(file_name)
145 elif command == 'help':
146     # Вывести справку о работе с программой.
147     print("Список команд:\n")
148     print("add - добавить работника;")
149     print("list - вывести список работников;")
150     print("select <стаж> - запросить работников со стажем;")
151     print("help - отобразить справку;")
152     print("load - загрузить данные из файла;")
153     print("save - сохранить данные в файл;")
154     print("exit - завершить работу с программой.")
155 else:
156     print(f"Неизвестная команда {command}", file=sys.stderr)
157
158
159 if __name__ == '__main__':
160     main()

```

Рисунок 1.7 – код программы ex3.py

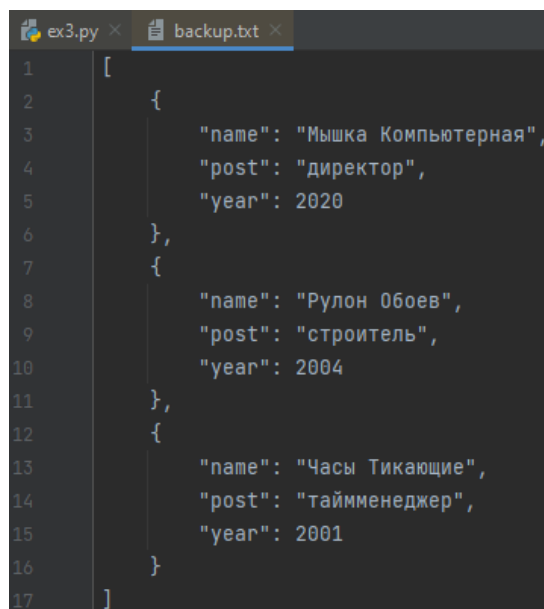
```

C:\Users\admin\anaconda3\python.exe C:/Users/admin/lab.2.16/example/ex3.py
>>> add
Фамилия и инициалы? Рулон Обоев
Должность? строитель
Год поступления? 2004
>>> add
Фамилия и инициалы? Часы Тикающие
Должность? таймменеджер
Год поступления? 2001
>>> add
Фамилия и инициалы? Мышка Компьютерная
Должность? иректорд
Год поступления? 2020
>>> lis
>>> Неизвестная команда lis
list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Мышка Компьютерная      |      иректорд      |      2020     |
+-----+-----+-----+-----+
|  2 | Рулон Обоев              |      строитель     |      2004     |
+-----+-----+-----+-----+
|  3 | Часы Тикающие            |      таймменеджер   |      2001     |
+-----+-----+-----+-----+
>>> save backup.txt
>>> exit

Process finished with exit code 0

```

Рисунок 1.8 – результат работы программы ex3.py



```

1  [
2      {
3          "name": "Мышка Компьютерная",
4          "post": "директор",
5          "year": 2020
6      },
7      {
8          "name": "Рулон Обоев",
9          "post": "строитель",
10         "year": 2004
11     },
12     {
13         "name": "Часы Тикающие",
14         "post": "таймменеджер",
15         "year": 2001
16     }
17 ]

```

Рисунок 1.9 – результат работы программы ex3.py

```

C:\Users\admin\anaconda3\python.exe C:/Users/admin/lab.2.16/example/ex3.py
>>> load backup.txt
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Мышка Компьютерная      |    директор    |   2020   |
+-----+-----+-----+-----+
|  2 | Рулон Обоев              |    строитель    |   2004   |
+-----+-----+-----+-----+
|  3 | Часы Тикающие            |  таймменеджер   |   2001   |
+-----+-----+-----+-----+
>>>

```

Рисунок 1.10 – результат работы программы ex3.py

## 2. Индивидуальное задание.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import json
6  import sys
7
8
9
10 def get_flight():
11     """
12     Запросить данные о полёте
13     """
14     flight_destination = input("Введите название пункта назначения ")
15     flight_number = input("Введите номер рейса ")
16     airplane_type = input("Введите тип самолета ")
17     return {
18         'flight_destination': flight_destination,
19         'flight_number': flight_number,
20         'airplane_type': airplane_type,
21     }
22
23
24 def display_flights(flights):
25     """
26     Отобразить список рейсов
27     """
28     if flights:
29         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
30             '-' * 4,
31             '-' * 30,
32             '-' * 20,
33             '-' * 15
34         )
35         print(line)
36         print(
37             '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
38                 "No",
39                 "Пункт назначения",
40                 "Номер рейса",
41                 "Тип самолета"

```

Рисунок 2.1 – код программы ind\_task.py

```

42         )
43     )
44     print(line)
45
46     for idx, flight in enumerate(flights, 1):
47         print(
48             '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
49                 idx,
50                 flight.get('flight_destination', ''),
51                 flight.get('flight_number', ''),
52                 flight.get('airplane_type', 0)
53             )
54         )
55     print(line)
56
57     else:
58         print("Список рейсов пуст")
59
60
61 def select_flights(flights, airplane_type):
62     """
63     Выбрать рейсы самолётов заданного типа
64     """
65     count = 0
66     res = []
67     for flight in flights:
68         if flight.get('airplane_type') == airplane_type:
69             count += 1
70             res.append(flight)
71     if count == 0:
72         print("рейсы не найдены")
73
74     return res
75
76
77 def save_workers(file_name, planes):
78     """
79     Сохранить всех работников в файл JSON.
80     """
81     # Открыть файл с заданным именем для записи.
82     with open(file_name, "w", encoding="utf-8") as fout:

```

Рисунок 2.2 – код программы ind\_task.py



```

83     # Выполнить сериализацию данных в формат JSON.
84     # Для поддержки кириллицы установим ensure_ascii=False
85     json.dump(planes, fout, ensure_ascii=False, indent=4)
86
87
88 def load_workers(file_name):
89     """
90     Загрузить всех работников из файла JSON.
91     """
92     # Открыть файл с заданным именем для чтения.
93     with open(file_name, "r", encoding="utf-8") as fin:
94         return json.load(fin)
95
96
97 def main():
98     """
99     Главная функция программы
100    """
101    flights = []
102    while True:
103        command = input(">>> ").lower()
104        if command == 'exit':
105            break
106
107        elif command == 'add':
108            flight = get_flight()
109            flights.append(flight)
110            if len(flights) > 1:
111                flights.sort(
112                    key=lambda item:
113                        item.get('flight_destination', ''))
114
115        elif command == 'list':
116            display_flights(flights)
117
118        elif command.startswith('select '):
119            parts = command.split(' ', maxsplit=1)
120            airplane_type = (parts[1].capitalize())
121            print(f"Для типа самолета {airplane_type}:")
122            selected = select_flights(flights, airplane_type)
123            display_flights(selected)

```

Рисунок 2.3 – код программы ind\_task.py

```

125 elif command == 'help':
126     # Вывести справку о работе с программой.
127     print("Список команд:\n")
128     print("add - добавить рейс;")
129     print("list - вывести список всех рейсов;")
130     print("select <тип самолета> - запросить рейсы указанного типа "
131           "самолета;")
132     print("help - отобразить справку;")
133     print("exit - завершить работу с программой.")
134
135 elif command.startswith("save "):
136     # Разбить команду на части для выделения имени файла.
137     parts = command.split(maxsplit=1)
138     # Получить имя файла.
139     file_name = parts[1]
140     # Сохранить данные в файл с заданным именем.
141     save_workers(file_name, flights)
142
143 elif command.startswith("load "):
144     # Разбить команду на части для выделения имени файла.
145     parts = command.split(maxsplit=1)
146     # Получить имя файла.
147     file_name = parts[1]
148     flights = load_workers(file_name)
149 else:
150     print(f"Неизвестная команда {command}", file=sys.stderr)
151
152
153 if __name__ == '__main__':
154     main()

```

Рисунок 2.4 – код программы ind\_task.py

```

C:\Users\admin\anaconda3\python.exe C:/Users/admin/lab.2.16/ind_task/ind_task.py
>>> add
Введите название пункта назначения Stavropol
Введите номер рейса 254
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Biisk
Введите номер рейса 938
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Krasnodar
Введите номер рейса 248
Введите тип самолета Passenger
>>> list
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Biisk | 938 | Passenger |
| 2 | Krasnodar | 248 | Passenger |
| 3 | Stavropol | 254 | Passenger |
+-----+-----+-----+-----+
>>> save backup.json
>>> exit

Process finished with exit code 0

```

Рисунок 2.5 – результат работы программы ind\_task.py

```
1 [
2   {
3     "flight_destination": "Biisk",
4     "flight_number": "938",
5     "airplane_type": "Passenger"
6   },
7   {
8     "flight_destination": "Krasnodar",
9     "flight_number": "248",
10    "airplane_type": "Passenger"
11  },
12  {
13    "flight_destination": "Stavropol",
14    "flight_number": "254",
15    "airplane_type": "Passenger"
16  }
17 ]
```

Рисунок 2.6 – файл backup.json

```
C:\Users\admin\anaconda3\python.exe C:/Users/admin/lab.2.16/ind_task/ind_task.py
>>> load backup.json
>>> list
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Biisk | 938 | Passenger |
| 2 | Krasnodar | 248 | Passenger |
| 3 | Stavropol | 254 | Passenger |
+-----+-----+-----+-----+
>>> |
```

Рисунок 2.7 – результат работы программы ind\_task.py

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

Функции для кодирования/декодирования данных:

## Quick Decoder Summary

<code>decode</code> (data[, maxdepth, some])	Decodes JSON5 serialized data from an <code>str</code> object.
<code>decode_latini</code> (data[, maxdepth, some])	Decodes JSON5 serialized data from a <code>bytes</code> object.
<code>decode_buffer</code> (obj[, maxdepth, some, wordlength])	Decodes JSON5 serialized data from an object that supports <code>read</code> and <code>readline</code> .
<code>decode_callback</code> (cb[, maxdepth, some, args])	Decodes JSON5 serialized data by invoking a callback function.
<code>decode_io</code> (fp[, maxdepth, some])	Decodes JSON5 serialized data from a file-like object.
<code>load</code> (fp, **kw)	Decodes JSON5 serialized data from a file-like object.
<code>loads</code> (s, *[, encoding])	Decodes JSON5 serialized data from a string.
<code>JsonSDecoderException</code> ([message, result])	Base class of any exception thrown by the parser.
<code>JsonSNestingTooDeep</code>	The maximum nesting level on the input data was exceeded.
<code>JsonSEOF</code>	The input ended prematurely.
<code>JsonSIllegalCharacter</code> ([message, result, ...])	An unexpected character was encountered.
<code>JsonSExtraData</code> ([message, result, character])	The input contained extraneous data.
<code>JsonSIllegalType</code> ([message, result, value])	The user supplied callback function returned illegal data.

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:

schema.json

```
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 }
```