

**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:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний

```
ex1.py x
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)
17
```

```
foo1.txt x  foo2.txt x
1  ["foo", "bar"]
```

```
foo1.txt x  foo2.txt x
1  ["foo", "bar"]
```

```
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)
17
```

```
ex2 x
↑ C:\Users\student-09-525\Desktop\LR-4\
↓ ['foo', 'bar']
⏮ ['foo', 'bar']
⏮
⏮ Process finished with exit code 0
```

```

ex3.py x
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",
42                 "Ф.И.О.",
43                 "Должность",
44                 "Год"
45             )
46         )
47         print(line)
48         # Вывести данные о всех сотрудниках.
49         for idx, worker in enumerate(staff, 1):
50             print(
51                 '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
52                     idx,
53                     worker.get('name', ''),
54                     worker.get('post', ''),
55                     worker.get('year', 0)

```

```
ex3.py x
55         worker.get('year', 0)
56     )
57 )
58     print(line)
59 else:
60     print("Список работников пуст.")
61
62
63 def select_workers(staff, period):
64     """
65     Выбрать работников с заданным стажем.
66     """
67     # Получить текущую дату.
68     today = date.today()
69     # Сформировать список работников.
70     result = []
71     for employee in staff:
72         if today.year - employee.get('year', today.year) >= period:
73             result.append(employee)
74     # Возвратить список выбранных работников.
75     return result
76
77
78 def save_workers(file_name, staff):
79     """
80     Сохранить всех работников в файл JSON.
81     """
82     # Открыть файл с заданным именем для записи.
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             return
111         elif command.startswith("load "):
112             file_name = command.split(" ", 1)[1]
113             workers = load_workers(file_name)
114         elif command.startswith("save "):
115             file_name = command.split(" ", 1)[1]
116             save_workers(file_name, workers)
117         elif command.startswith("select "):
118             file_name, period = command.split(" ", 2)
119             workers = select_workers(load_workers(file_name), int(period))
120         elif command == "list":
121             print(workers)
122         elif command == "help":
123             print("Возможные команды: list, save, load, select, help, exit")
124         else:
125             print("Неизвестная команда")
126
127 if __name__ == '__main__':
128     main()
```

```
ex3.py x
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.
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()
161
```

```
ex3 x
C:\Users\student-09-525\Desktop\LR-4\venv\Scripts\python.exe C:/Users/student-0
>>> add
Фамилия и инициалы? Efgfh Dh
Должность? Ed
Год поступления? 2012
>>> add
Фамилия и инициалы? Ksks TR
Должность? Dbd
Год поступления? 2002
>>> add
Фамилия и инициалы? jjd dd
Должность? Di
Год поступления? 2020
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Efgfh Dh                  | Ed                  |  2012 |
+-----+-----+-----+-----+
|  2 | Ksks TR                   | Dbd                 |  2002 |
+-----+-----+-----+-----+
|  3 | jjd dd                    | Di                  |  2020 |
+-----+-----+-----+-----+
>>> select 10
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Efgfh Dh                  | Ed                  |  2012 |
+-----+-----+-----+-----+
|  2 | Ksks TR                   | Dbd                 |  2002 |
+-----+-----+-----+-----+
>>> save backup.txt
>>> exit

Process finished with exit code 0
```

```
ex3.py x backup.txt x
1  [
2      {
3          "name": "Efgfh Dh",
4          "post": "Ed",
5          "year": 2012
6      },
7      {
8          "name": "Ksks TR",
9          "post": "Dbd",
10         "year": 2002
11     },
12     {
13         "name": "jjd dd",
14         "post": "Di",
15         "year": 2020
16     }
17 ]
```

```
ex3 x
C:\Users\student-09-525\Desktop\LR-4\venv\Scripts\python.exe C:/Users/student-
>>> load backup.txt
>>> list
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Efgfh Dh | Ed | 2012 |
+-----+-----+-----+-----+
| 2 | Ksks TR | Dbd | 2002 |
+-----+-----+-----+-----+
| 3 | jjd dd | Di | 2020 |
+-----+-----+-----+-----+
>>> |
```

2. Индивидуальное задание (рис. 1-6).


```

idz.py × backup.txt ×
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import json
6      import sys
7
8
9      def get_flight():
10         """
11         Запросить данные о полёте
12         """
13         flight_destination = input("Введите название пункта назначения ")
14         flight_number = input("Введите номер рейса ")
15         airplane_type = input("Введите тип самолета ")
16         return {
17             'flight_destination': flight_destination,
18             'flight_number': flight_number,
19             'airplane_type': airplane_type,
20         }
21
22
23     def display_flights(flights):
24         """
25         Отобразить список рейсов
26         """
27         if flights:
28             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
29                 '-' * 4,
30                 '-' * 30,
31                 '-' * 20,
32                 '-' * 15
33             )
34             print(line)
35             print(
36                 '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
37                     "No",
38                     "Пункт назначения",
39                     "Номер рейса",
40                     "Тип самолета"
41                 )
42             )
43             print(line)
44
45             for idx, flight in enumerate(flights, 1):
46                 print(
47                     '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
48                         idx,
49                         flight.get('flight_destination', ''),
50                         flight.get('flight_number', ''),
51                         flight.get('airplane_type', '')
52                     )
53                 )
54             print(line)
55

```

Рисунок 1 – Код программы

```

55
56     else:
57         print("Список рейсов пуст")
58
59
60 def select_flights(flights, airplane_type):
61     """
62     Выбрать рейсы самолётов заданного типа
63     """
64     count = 0
65     res = []
66     for flight in flights:
67         if flight.get('airplane_type') == airplane_type:
68             count += 1
69             res.append(flight)
70     if count == 0:
71         print("рейсы не найдены")
72
73     return res
74
75
76 def save_workers(file_name, planes):
77     """
78     Сохранить всех работников в файл JSON.
79     """
80     # Открыть файл с заданным именем для записи.
81     with open(file_name, "w", encoding="utf-8") as fout:
82         # Выполнить сериализацию данных в формат JSON.
83         # Для поддержки кириллицы установим ensure_ascii=False
84         json.dump(planes, fout, ensure_ascii=False, indent=4)
85
86
87 def load_workers(file_name):
88     """
89     Загрузить всех работников из файла JSON.
90     """
91     # Открыть файл с заданным именем для чтения.
92     with open(file_name, "r", encoding="utf-8") as fin:
93         return json.load(fin)
94
95
96 def main():
97     """
98     Главная функция программы
99     """
100     flights = []
101     while True:
102         command = input(">>> ").lower()
103         if command == 'exit':
104             break
105
106         elif command == 'add':
107             flight = get_flight()
108             flights.append(flight)
109             if len(flights) > 1:
load_workers()

```

Рисунок 2 – Код программы, продолжение

```

109         if len(flights) > 1:
110             flights.sort(
111                 key=lambda item:
112                     item.get('flight_destination', ''))
113
114     elif command == 'list':
115         display_flights(flights)
116
117     elif command.startswith('select '):
118         parts = command.split(' ', maxsplit=1)
119         airplane_type = (parts[1].capitalize())
120         print(f"Для типа самолета {airplane_type}:")
121         selected = select_flights(flights, airplane_type)
122         display_flights(selected)
123
124     elif command == 'help':
125         # Вывести справку о работе с программой.
126         print("Список команд:\n")
127         print("add - добавить рейс;")
128         print("list - вывести список всех рейсов;")
129         print("select <тип самолета> - запросить рейсы указанного типа "
130             "самолета;")
131         print("help - отобразить справку;")
132         print("exit - завершить работу с программой.")
133
134     elif command.startswith("save "):
135         # Разбить команду на части для выделения имени файла.
136         parts = command.split(maxsplit=1)
137         # Получить имя файла.
138         file_name = parts[1]
139         # Сохранить данные в файл с заданным именем.
140         save_workers(file_name, flights)
141
142     elif command.startswith("load "):
143         # Разбить команду на части для выделения имени файла.
144         parts = command.split(maxsplit=1)
145         # Получить имя файла.
146         file_name = parts[1]
147         # Сохранить данные в файл с заданным именем.
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()

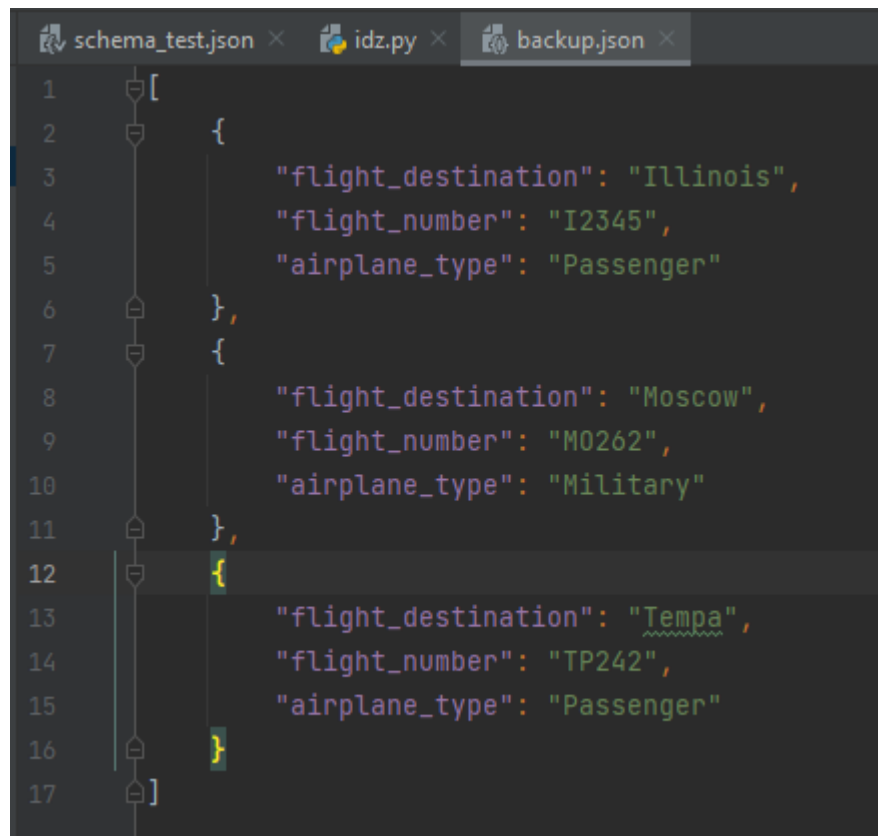
```

Рисунок 3 – Код программы, продолжение

```
idz x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Evil/PycharmProj
>>> add
Введите название пункта назначения Tempa
Введите номер рейса TP242
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Illinois
Введите номер рейса I2345
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Moscow
Введите номер рейса M0262
Введите тип самолета Military
>>> list
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Illinois | I2345 | Passenger |
| 2 | Moscow | M0262 | Military |
| 3 | Tempa | TP242 | Passenger |
+-----+-----+-----+-----+
>>> save backup.json
>>> exit

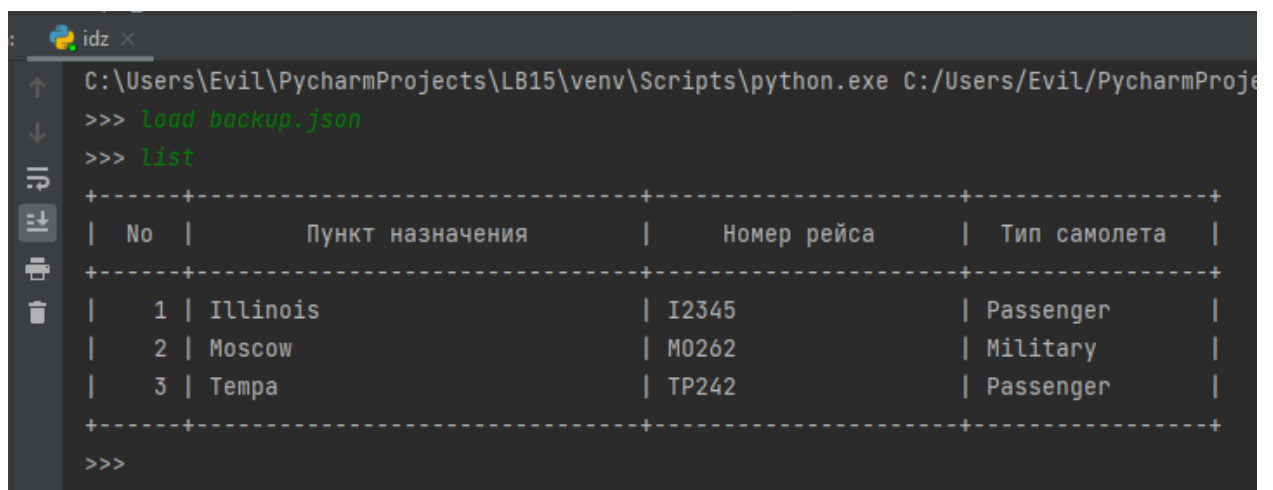
Process finished with exit code 0
```

Рисунок 4 – Код программы, продолжение



```
1  [
2      {
3          "flight_destination": "Illinois",
4          "flight_number": "I2345",
5          "airplane_type": "Passenger"
6      },
7      {
8          "flight_destination": "Moscow",
9          "flight_number": "M0262",
10         "airplane_type": "Military"
11     },
12     {
13         "flight_destination": "Tempa",
14         "flight_number": "TP242",
15         "airplane_type": "Passenger"
16     }
17 ]
```

Рисунок 5 – Содержимое файла backup.json



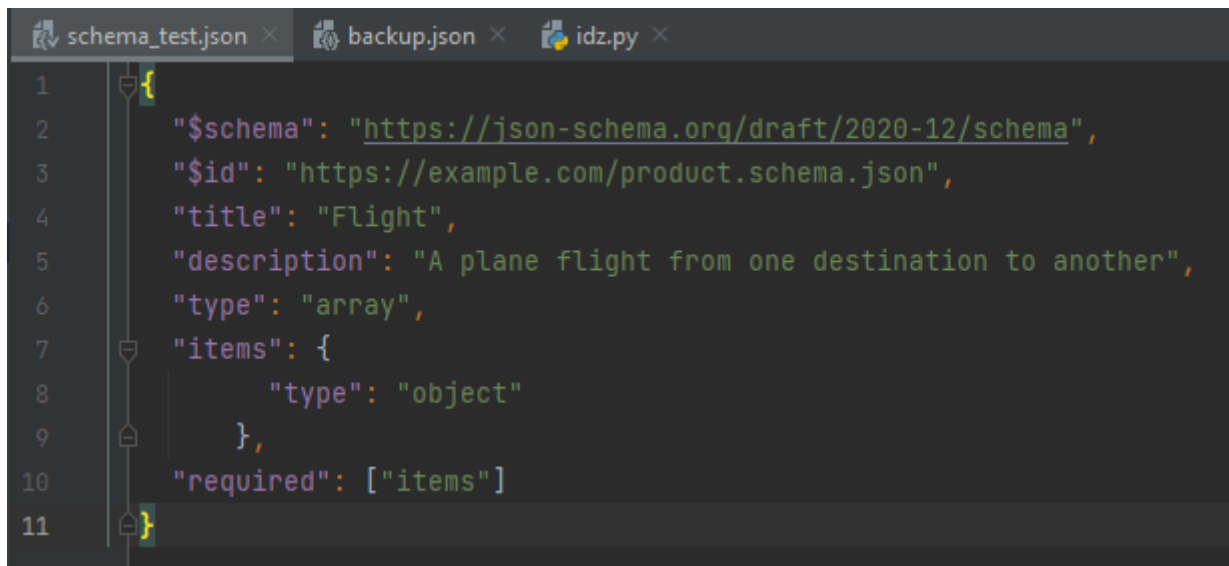
```
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Evil/PycharmProje
>>> load backup.json
>>> list
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Illinois	I2345	Passenger
2	Moscow	M0262	Military
3	Tempa	TP242	Passenger

```
>>>
```

Рисунок 6 – Работа функции загрузки из файла

Задание повышенной сложности. JSON Schema. (рис. 7).

A screenshot of a code editor with three tabs: 'schema_test.json', 'backup.json', and 'idz.py'. The 'schema_test.json' tab is active, showing a JSON schema for 'Flight'. The schema is a JSON object with the following properties: '\$schema' pointing to 'https://json-schema.org/draft/2020-12/schema', '\$id' pointing to 'https://example.com/product.schema.json', 'title' set to 'Flight', 'description' set to 'A plane flight from one destination to another', 'type' set to 'array', 'items' set to an object with 'type' set to 'object', and 'required' set to an array containing 'items'. The code is numbered from 1 to 11 on the left margin.

```
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": "array",  
7   "items": {  
8     "type": "object"  
9   },  
10  "required": ["items"]  
11 }
```

Рисунок 7 – Схема данных для индивидуального задания

3. 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 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</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>Json5EncoderException</code>	Base class of any exception thrown by the serializer.
<code>Json5UnstringifiableType</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_latin1</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 the buffer interface.
<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>Json5DecoderException</code> ([message, result])	Base class of any exception thrown by the parser.
<code>Json5NestingTooDeep</code>	The maximum nesting level on the input data was exceeded.
<code>Json5EOF</code>	The input ended prematurely.
<code>Json5IllegalCharacter</code> ([message, result, ...])	An unexpected character was encountered.
<code>Json5ExtraData</code> ([message, result, character])	The input contained extraneous data.
<code>Json5IllegalType</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 x
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 }
```