

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Анализ данных»
Вариант 28

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Работа с данными формата JSON в языке Python.

Цель: приобрести навыки работы с данными формата JSON с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Выполнен пример лабораторной работы – модифицирован пример №1 из Лабораторной работы №11 дисциплины “Программирование на Python”: в примере создаётся словарь, хранящий информацию о сотрудниках – ФИО, должность, год поступления на работу. Обработываемые команды: вывод всех сотрудников (list), вывод сводки о имеющихся командах (help), добавление сотрудника (add), вывод сотрудников относительно требуемой продолжительности работы (select “срок”), выход из программы (exit). Теперь данные о внесённых в список людях сохраняются и могут быть считаны в файл формата JSON (Команды save и load соответственно).

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import json
5  import sys
6  from datetime import date
7
8
9  def get_worker():
10     """
11     Запросить данные о работнике.
12     """
13     name = input("Фамилия и инициалы? ")
14     post = input("Должность? ")
15     year = int(input("Год поступления? "))
16
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         )
```

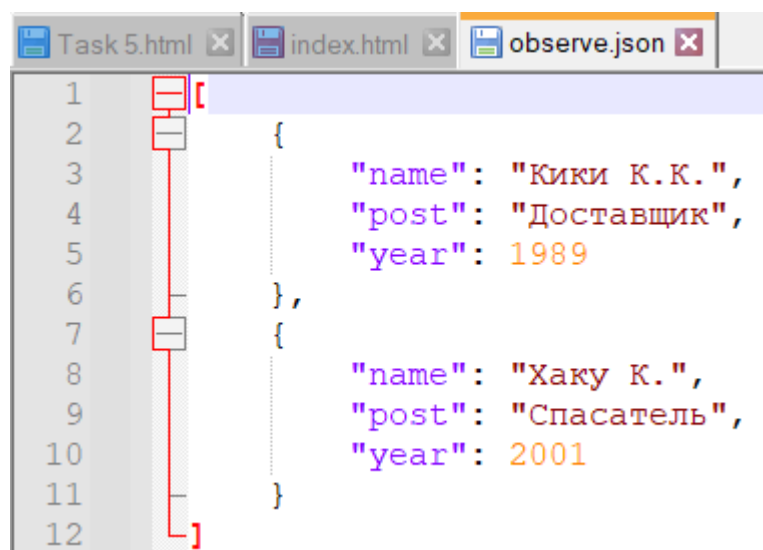
Рисунок 1. Код примера

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_2\Программы и файлы>python Example_1.py
>>> list
Список работников пуст.
>>> add
Фамилия и инициалы? Кики К.К.
Должность? Доставщик
Год поступления? 1989
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Кики К.К. | Доставщик | 1989 |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Хаку К.
Должность? Спасатель
Год поступления? 2001
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Кики К.К. | Доставщик | 1989 |
| 2 | Хаку К. | Спасатель | 2001 |
+-----+-----+-----+-----+
>>> save Observe.json

```

Рисунок 2. Выполнение программы и сохранение данных



```

1  [
2
3      {
4          "name": "Кики К.К.",
5          "post": "Доставщик",
6          "year": 1989
7      },
8      {
9          "name": "Хаку К.",
10         "post": "Спасатель",
11         "year": 2001
12     }
13 ]

```

Рисунок 3. Сохранённый список

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_2\Программы и файлы>python Example_1.py
>>> list
Список работников пуст.
>>> load Observe.json
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Кики К.К. | Доставщик | 1989 |
| 2 | Хаку К. | Спасатель | 2001 |
+-----+-----+-----+-----+
>>>

```

Рисунок 4. Получение данных из файла

2. Выполнено индивидуальное задание – модифицировано задание №9 из Лабораторной работы №11 дисциплины “Программирование на Python”: в задании создаётся словарь, хранящий информацию о людях – фамилия, имя, телефонный номер, дата рождения. Требовалось обрабатывать несколько команд – вывод всех людей (list), вывод сводки о имеющихся

командах (help), добавление человека (add), вывод людей относительно требуемого месяца рождения (select “месяц”), выход из программы (exit). Теперь данные о внесённых в список людях сохраняются и могут быть считаны в файл формата JSON (Команды save и load соответственно).

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import json
5 import sys
6
7
8 def new_human():
9     """Запросить данные о человеке."""
10
11     name = input("Name - ")
12     surname = input("Surname - ")
13     telephone = input("Telephone number - ")
14     happy_birthday = input("Date of birth (Day.Month.Year) - ")
15     # Создать словарь.
16     return {
17         "name": name,
18         "surname": surname,
19         "telephone": telephone,
20         "birthday": happy_birthday
21     }
22
23
24 def display_people(people):
25     """Отобразить список людей."""
26
27     if people:
28         # Заголовок таблицы.
29
30         line = "|-{}-{}-{}-{}-{}-{}-|".format(
31             "-" * 5, "-" * 25, "-" * 25, "-" * 25, "-" * 25, "-" * 18)
32         print(line)
33         print("| {:^5} | {:^24} | {:^25} | {:^25} | {:^18} |".format(
34             "№", "Name", "Surname", "Telephone", "Birthday"))
35         print(line)
36         for number, human in enumerate(people, 1):
37             print("| {:^5} | {:<24} | {:<25} | {:<25} | {:<18} |".format(number, human.get(

```

Рисунок 5. Код индивидуального задания

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data_Analysis_2\Программы и файлы>python Task_1.py
Good day!, please, enter your command: ('help' will list all of them)
>>> add
Name - Suzuki
Surname - Satoru
Telephone number - 40000000004
Date of birth (Day.Month.Year) - 07.07.2015
>>> add
Name - Alebrije
Surname - Wisdom
Telephone number - 99999999999
Date of birth (Day.Month.Year) - 01.11.2007
>>> list
+-----+-----+-----+-----+-----+
| № | Name | Surname | Telephone | Birth |
+-----+-----+-----+-----+-----+
| 1 | Suzuki | Satoru | 40000000004 | 07.07.2015 |
| 2 | Alebrije | Wisdom | 99999999999 | 01.11.2007 |
+-----+-----+-----+-----+-----+
>>> save Purgatory.json
```

Рисунок 6. Выполнение программы и сохранение результата

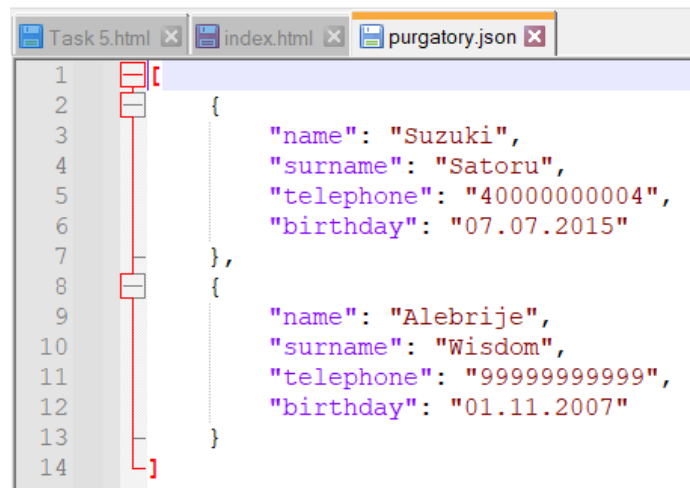


Рисунок 7. Сохранённый список

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_2\Программы и файлы>python Task_1.py
Good day!, please, enter your command: ('help' will list all of them)
>>> list
There are no people in list!
>>> load Purgatory.json
>>> list

```

№	Name	Surname	Telephone	Birthday
1	Suzuki	Satoru	40000000004	07.07.2015
2	Alebrije	Wisdom	99999999999	01.11.2007

Рисунок 8. Получение данных из файла

3. Выполнено задание повышенной сложности – модифицированы выполненные ранее пример и индивидуальное задание (При помощи jsonscheme производится проверка содержимого json файлов. Если они эквиваленты предоставленной программе схеме, загрузка выполняется).

```

78
79 def load_people(file_name):
80     """Загрузить всех людей из файла JSON."""
81
82     # Открыть файл с заданным именем для чтения.
83     with open(file_name, "r", encoding="utf-8") as fin:
84         loaded = json.load(fin)
85     try:
86         jsonschema.validate(loaded, schema)
87         print(">>> Data is obtained!")
88     except jsonschema.exceptions.ValidationError as e:
89         print(">>> Data's structure is invalid. Please, check your JSON file.Error:")
90         print(e.message) # Ошибка валидации будет выведена на экран
91     return loaded
92
93
94 def main():
95     """Главная функция программы."""
96
97     print("Good day!, please, enter your command: ('help' will list all of them)")
98     people = []
99     while True:
100         command = input(">>> ").lower()
101         if command == "exit":
102             break
103         elif command == "add":
104             human = new_human()
105             people.append(human)
106             # Сортировка идёт по фамилиям. Если фамилии одинаковые, то рассматриваются имена
107             people.sort(key=lambda person: (person.get("surname", ""), person.get("name", "")))
108         elif command == "list":
109             display_people(people)
110         elif command.startswith("select "):
111             which_month = command.split(" ", maxsplit=1)
112             user_month = int(which_month[1])
113             camp_month = select_people(people, user_month)
114

```

Рисунок 9. Изменённый код примера

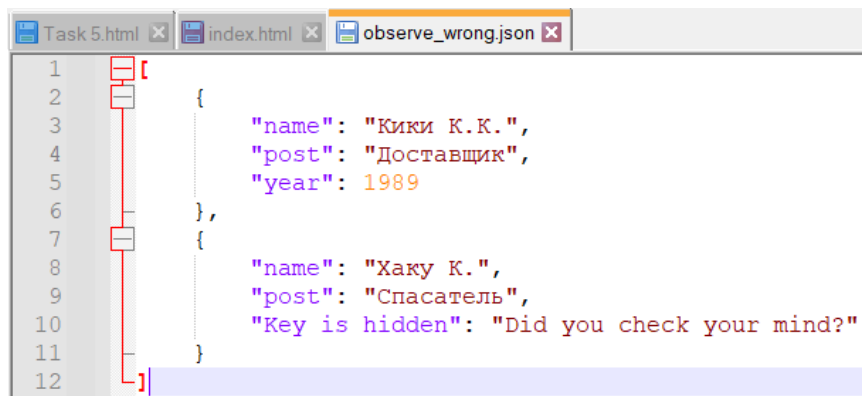


Рисунок 10. Изменённое содержимое json файла

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_2\Программы и файлы>python Example_1_checking_json.py
>>> list
Список работников пуст.
>>> load observe_wrong.json
>>> Data's structure is invalid. Please, check your JSON file.Error:
'year' is a required property
>>> load observe.json
>>> Data is obtained!
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Кики К.К. | Доставщик | 1989 |
| 2 | Хаку К. | Спасатель | 2001 |
+-----+-----+-----+-----+

```

Рисунок 11. Попытка загрузить неверный json файл для примера

```

78 def load_workers(file_name):
79     """
80     Загрузить всех работников из файла JSON.
81     """
82     # Открыть файл с заданным именем для чтения.
83     with open(file_name, "r", encoding="utf-8") as fin:
84         loaded = json.load(fin)
85     try:
86         jsonschema.validate(loaded, schema)
87         print(">>> Data is obtained!")
88     except jsonschema.exceptions.ValidationError as e:
89         print(">>> Data's structure is invalid. Please, check your JSON file.Error:")
90         print(e.message) # Ошибка валидации будет выведена на экран
91     return loaded
92
93
94 def main():
95     """
96     Главная функция программы.
97     """
98     # Список работников.
99     workers = []
100     # Организовать бесконечный цикл запроса команд.
101     while True:
102         # Запросить команду из терминала.
103         command = input(">>> ").lower()
104         # Выполнить действие в соответствие с командой.
105         if command == "exit":
106             break
107         elif command == "add":
108             # Запросить данные о работнике.
109             worker = get_worker()
110             # Добавить словарь в список.
111             workers.append(worker)
112             # Отсортировать список в случае необходимости.
113             if len(workers) > 1:

```

Рисунок 12. Изменённый код индивидуального задания

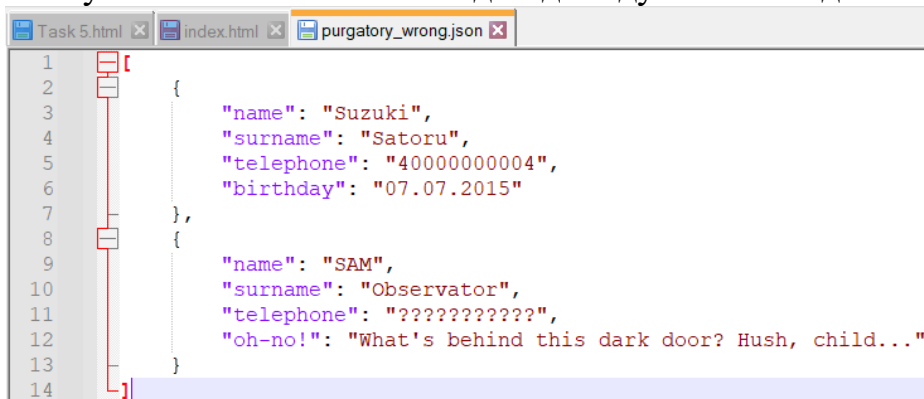


Рисунок 13. Изменённое содержимое json файла

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_2\Программы и файлы>python Task_1_checking_json.py
Good day!, please, enter your command: ('help' will list all of them)
>>> list
There are no people in list!
>>> load purgatory_wrong.json
>>> Data's structure is invalid. Please, check your JSON file.Error:
'birthday' is a required property
>>> load purgatory.json
>>> Data is obtained!
>>> list
```

№	Name	Surname	Telephone	Birthday
1	Suzuki	Satoru	40000000004	07.07.2015
2	Alebrije	Wisdom	99999999999	01.11.2007

Рисунок 14. Попытка загрузить неверный json файл для задания 4.

1) Для чего используется JSON?

Ответ: JSON - текстовый формат обмена данными, основанный на JavaScript. JSON легко читаемый, его формат был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате 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). Легко читаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента.

2) Какие типы значений используются в JSON?

Ответ: В качестве значений в JSON могут быть использованы: запись (неупорядоченное множество пар ключ-значение, заключённое в фигурные скобки «{ }»). Описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми), массив (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 (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?

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

4) В чём отличие формата данных JSON5 от JSON?

Ответ: JSON5 — это расширение стандарта JSON, которое повышает читаемость и удобство написания JSON-данных. Главные отличия JSON5: допустимы комментарии, необязательно использовать кавычки для ключей (только если ключ состоит из букв, цифр или знаков подчёркивания и не является зарезервированным словом), есть специальный формат для дат и времени, поддерживает многострочный текст (это позволяет записывать строки без неудобного экранирования), допускает запись чисел с подчёркиваниями для улучшения читаемости, поддерживает шестнадцатеричную и восьмеричную системы счисления.

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

Ответ: *json5* – библиотека Python для работы с данными в формате JSON5. Она предоставляет функции для чтения и записи данных в/из формата JSON5. Команды: *json5.load(text)*, *json5.dump(text)*.

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

Ответ: *json.dump()* - конвертировать python объект в json и записать в файл (*json.dumps()* - тоже самое, но в строку).

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

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

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

Ответ: `json.load()` - прочитать json из файла и конвертировать в python объект (`json.loads()` - тоже самое, но из строки с json (s на конце от string/строка)).

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

Ответ: 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.

10) Что такое схема данных? Приведите схему данных для примера 1.

Ответ: JSON Schema — это распространенный стандарт описания структуры данных. Спецификация стандарта и популярные сценарии его использования доступны на ресурсе <http://json-schema.org/>. Схема создана для описания JSON-данных, но и сама она при этом является JSON-объектом. С помощью ключевых слов в схеме создаются правила валидации структуры объекта и типов его полей. Для примера №1 схема выглядит:

```
schema = {  
    "type": "array",  
    "items": {  
        "type": "object",  
        "properties": {  
            "name": {"type": "string"},  
            "post": {"type": "string"},  
            "year": {"type": "integer"}  
        },  
    },  
}
```

```
"required": ["name", "post", "year"]  
}  
}
```

Вывод: в ходе выполнения лабораторной работы, приобретены навыки работы с данными формата JSON с помощью языка программирования Python версии 3.x.