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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.16 Работа с данными формата JSON в языке Python»

ОТЧЕТ
по лабораторной работе №19
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 2.16 Работа с данными формата JSON в языке Python

Цель работы: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

Ход работы:

Индивидуальные задания

Задание

Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON. *Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.*

Задание повышенной сложности

Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. В следствие чего, необходимо после загрузки из файла JSON выполнять валидацию загруженных данных. Валидацию данных необходимо производить с использованием спецификации JSON Schema, описанной на сайте https://json-schema.org/. Одним из возможных вариантов работы с JSON Schema является использование пакета `jsonschema`, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

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

import sys
import json
import jsonschema
from jsonschema import validate


schema = {
    "type": "object",
    "properties": {
        "product": {
            "type": "string"
        },
        "shop": {
            "type": "string"
        },
        "cost": {
            "type": "number"
        }
    }
}
```

```python
def add_product():
    """
    Ввод информации о товарах.
    """
    prod = input("Введите название товара: ")
    shop = input("Введите название магазина: ")
    cost = float(input("Введите стоимость товара: "))

    return {
        'product': prod,
        'shop': shop,
        'cost': cost
    }


def save_products(file_name, products):
    """
    Сохранить список всех товаров в формате JSON
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(products, fout, ensure_ascii=False, indent=4)


def load_products(file_name):
    """
    Загрузить список всех товаров из файла JSON
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def product_list(products):
    """
    Вывод списка товаров
    """
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20
    )
    print(line)
    print(
        '| {:^25} | {:^15} | {:^14} |'.format(
            "Товар",
            "Магазин",
            "Стоимость"
        )
    )
    print(line)

    for product in products:
        print(
            '| {:^25} | {:^15} | {:^14} |'.format(
                product.get('product', ''),
                product.get('shop', ''),
                product.get('cost', 0)
            )
        )
    print(line)


def select(products, shop):
    """
    Выбрать товары из конкретного магазина.
```

```
    """
    result = []
    for product in products:
        if product.get('shop', '') == shop:
            result.append(product)
    return result


def get_help():
    print("Список команд:\n")
    print("add - добавить информацию о товаре;")
    print("list - вывести список товаров;")
    print("select - запросить товары из одного магазина;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    print("load - загрузить данные из файла;")
    print("save - сохранить данные в файл;")


def error(command):
    print(f"Неизвестная команда {command}", file=sys.stderr)


def validation(json_data):
    """
    Валидация данных
    """
    try:
        validate(instance=json_data, schema=schema)
    except:
        raise jsonschema.exceptions.ValidationError("Данные недействительны")

    msg = "Данные успешно загружены"
    return True, msg


def main():
    """
    Главная функция программы.
    """
    products = []

    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            product = add_product()
            products.append(product)
            if len(products) > 1:
                products.sort(key=lambda item: item.get('shop', ''))

        elif command == 'list':
            product_list(products)

        elif command == 'select':
            sel_shop = input("Введите магазин: ")
            selected = select(products, sel_shop)
            product_list(selected)

        elif command == 'help':
            get_help()
```

```
        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_products(file_name, products)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            products = load_products(file_name)
            for prod in products:
                is_valid, msg = validation(prod)
                print(prod)
                print(msg)

        else:
            error(command)


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

Рисунок 1 – Код программы для 1 индивидуального задания

```
>>> load testfile.json
{'product': 'кола', 'shop': 'магнит', 'cost': 100.0}
Данные успешно загружены
{'product': 'макароны', 'shop': 'пятерочка', 'cost': 44.0}
Данные успешно загружены
>>> load testfile2.json
{'product': 'кола', 'shop': 'магнит', 'cost': 100.0}
Данные успешно загружены
Traceback (most recent call last):
  File "E:\GitHub\laba19\individualShema.py", line 174, in <module>
    main()
  File "E:\GitHub\laba19\individualShema.py", line 165, in main
    is_valid, msg = validation(prod)
  File "E:\GitHub\laba19\individualShema.py", line 120, in validation
    raise jsonschema.exceptions.ValidationError("Данные недействительны")
jsonschema.exceptions.ValidationError: Данные недействительны

Process finished with exit code 1
```

Рисунок 2 – Результат работы программы

Вывод: были приобретены навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы

1. Для чего используется JSON? JSON (англ. JavaScript Object Notation, обычно произносится как /ˈdʒeɪsən/ JAY-sən) – текстовый формат обмена данными, основанный на JavaScript.

2. Какие типы значений используются в JSON?

– 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? 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. 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.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5? Сериализация данных в формат JSON, десериализация данных из формата JSON.

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

1. **Сериализация** данных в формат JSON:

```
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()?

1. **Сериализация** данных в формат JSON:

```
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?

2. **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)
```

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

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema?

Что такое схема данных? Схема JSON — это декларативный язык, который позволяет аннотировать и проверять документы JSON.

Приведите схему данных для примера 1.

```
schema = {
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "post": {
            "type": "string"
        },
        "year": {
            "type": "number"
        }
    }
}
```