МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.16

Работа сданными формата JSON в языке Python.

Выполнил студент группы

ИВТ-б-о-21-1

Мальцев Н.А. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь 2022

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

Порядок выполнения работы:

Проработка примера:

Код программы:

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

import json
import sys
from datetime import date


def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))
    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }


def display_workers(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
        print(line)
    else:
```

```
        print("Список работников пуст.")


def select_workers(staff, period):

    today = date.today()
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)
    return result


def save_workers(file_name, staff):

    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_workers(file_name):
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main():

    workers = []
    while True:
        command = input(">>> ").lower()
        if command == "exit":
            break

        elif command == "add":
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))
        elif command == "list":
            display_workers(workers)
        elif command.startswith("select "):
            parts = command.split(maxsplit=1)
            period = int(parts[1])
            selected = select_workers(workers, period)
            display_workers(selected)
        elif command.startswith("save "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            save_workers(file_name, workers)
        elif command.startswith("load "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            workers = load_workers(file_name)
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
```

```
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("load - загрузить данные из файла;")
            print("save - сохранить данные в файл;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


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

Результат выполнения программы:

```
>>> save primer
>>> load primer
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Богадуров В.И.                 | Заместитель директора |     2021 |
|    2 | Мальцев Н.А.                   | Директор             |     2020 |
+------+--------------------------------+----------------------+----------+
>>> load primer
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Богадуров В.И.                 | Заместитель директора |     2021 |
|    2 | Мальцев Н.А.                   | Директор             |     2020 |
|    3 | Сартр Ж.П.                     | Маркетолог           |     2022 |
+------+--------------------------------+----------------------+----------+
>>>
```

Рисунок 1. Результат работы программы

Выполнение задания:

Код программы:

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

import sys
import json
import datetime


def add(list_man):
    # Запросить данные .
    name = input("Имя:  ")
    number = input("Номер телефона ")
    date_ = input("Дата рождения: ")
    date_ = datetime.datetime.strptime(date_, "%Y-%m-%d").date()

    # Создать словарь.
    man = {
        "name": name,
        "number": number,
        "date": date_,
    }

    # Добавить словарь в список.
    list_man.append(man)
    # Отсортировать список.
    if len(list_man) > 1:
```

```
        list_man.sort(key=lambda item: item.get("date", ""))
    return list_man


def list_d(list_man):
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-
" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20}  |".format(
                    idx, man.get("name", ""), man.get("number", ""),
man.get("date", "")
                )
            )

        print(line)
    else:
        print("Список работников пуст: ")


def select(command_d, mans_list):
    parts_ = command_d.split(" ", maxsplit=1)
    sel = parts_[1]
    count = 0
    for man in mans_list:
        if man.get("number") == sel:
            count += 1
            print("{:>4}: {}".format(count, man.get("name", "")))
            print("Номер телефона:", man.get("number", ""))
            print("Дата рождения:", man.get("date", ""))

    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")


def save_workers(file_name_1, staff):

    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)


def load_workers(file_name_2):
    with open(file_name_2, "r", encoding="utf-8") as fin:
        return json.load(fin)


def help_d():
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить человека;")
    print("list - вывести список людей;")
    print("select <товар> - информация о человеке;")
```

```
    print("save <имя файла> - сохранение данных в файл")
    print("losd <имя файла> - загрузка данных из файла")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")


if __name__ == "__main__":
    manlist = []
    while True:
        # Запросить команду
        command = input(">>> ").lower()
        if command == "exit":
            break
        elif command == "add":
            manlist = add(manlist)
        elif command == "list":
            list_d(manlist)
        elif command.startswith("select "):
            select(command, manlist)
        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_workers(file_name, manlist)
        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            manlist = load_workers(file_name)
        elif command == "help":
            help_d()
        else:
            print("неизвестная команда {command}", file=sys.stderr)
```

Результат выполнения программы:

```
>>> save indtask
>>> load indtask
>>> list
+------+------------------------------+--------------------+----------------------+
|  No  |             Имя              |    Номер телефона  |    Дата рождения     |
+------+------------------------------+--------------------+----------------------+
|    1 | Jason                        | 837402234          | 2002-01-22           |
|    2 | Nik                          | 893749233          | 2003-02-23           |
+------+------------------------------+--------------------+----------------------+
>>>
```

Рисунок 2. Результат выполнения

JSON-файл:

```
[
    {
        "name": "Jason",
        "number": "837402234",
        "date": "2002-01-22"
    },
    {
        "name": "Nik",
        "number": "893749233",
        "date": "2003-02-23"
    }
]
```

Рисунок 3. Содержание json-файла

Задание повышенной сложности:

Код программы:

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

import sys
import json
import datetime
import jsonschema


def add(list_man):
    # Запросить данные .
    name = input("Имя:  ")
    number = input("Номер телефона ")
    date_ = input("Дата рождения: ")
    date_ = datetime.datetime.strptime(date_, "%Y-%m-%d").date()

    # Создать словарь.
    man = {
        "name": name,
        "number": number,
        "date": date_,
    }

    # Добавить словарь в список.
    list_man.append(man)
    # Отсортировать список.
    if len(list_man) > 1:
        list_man.sort(key=lambda item: item.get("date", ""))
    return list_man


def list_d(list_man):
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-
" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20}  |".format(
                    idx, man.get("name", ""), man.get("number", ""),
man.get("date", "")
                )
            )

        print(line)
    else:
        print("Список работников пуст: ")


def select(command_d, mans_list):
```

```
    parts_ = command_d.split(" ", maxsplit=1)
    sel = parts_[1]
    count = 0
    for man in mans_list:
        if man.get("number") == sel:
            count += 1
            print("{:>4}: {}".format(count, man.get("name", "")))
            print("Номер телефона:", man.get("number", ""))
            print("Дата рождения:", man.get("date", ""))

    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")


def save_workers(file_name_1, staff):
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)


def load_workers(file_name_2):
    schema = {
        "type": "array",
        "items": [
            {
                "type": "object",
                "properties": {
                    "name": {"type": "string"},
                    "number": {"type": "string"},
                    "date": {"type": "string"},
                },
                "required": ["name", "number", "date"],
            },
            {
                "type": "object",
                "properties": {
                    "name": {"type": "string"},
                    "number": {"type": "string"},
                    "date": {"type": "string"},
                },
                "required": ["name", "number", "date"],
            },
        ],
    }

    with open(file_name_2, "r", encoding="utf-8") as fin:
        loadf = json.load(fin)
        validator = jsonschema.Draft7Validator(schema)
        try:
            if not validator.validate(loadf):
                print("Валидация прошла успешно")
        except jsonschema.exceptions.ValidationError:
            print("Ошибка валидации", file=sys.stderr)
            exit()
        return loadf


def help_d():
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить человека;")
    print("list - вывести список людей;")
    print("select <товар> - информация о человеке;")
    print("save <имя файла> - сохранение данных в файл")
```

```
    print("losd <имя файла> - загрузка даннных из файла")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")


if __name__ == "__main__":
    manlist = []
    while True:
        # Запросить команду
        command = input(">>> ").lower()
        if command == "exit":
            break
        elif command == "add":
            manlist = add(manlist)
        elif command == "list":
            list_d(manlist)
        elif command.startswith("select "):
            select(command, manlist)
        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_workers(file_name, manlist)
        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            manlist = load_workers(file_name)
        elif command == "help":
            help_d()
        else:
            print("неизвестная команда {command}", file=sys.stderr)
```

Результат выполнения программы:

```
"C:\Users\Николай Мальцев\AppData\Local\Programs\Python\Python311\python.exe" "C:/Users/Ник
>>> load indtask
Валидация прошла успешно
>>> list
+------+--------------------------------+----------------------+----------------------+
|  No  |              Имя               |    Номер телефона    |    Дата рождения     |
+------+--------------------------------+----------------------+----------------------+
|    1 | Jason                          | 837402234            | 2002-01-22           |
|    2 | Nik                            | 893749233            | 2003-02-23           |
+------+--------------------------------+----------------------+----------------------+
>>>
```

Рисунок 4.

Ответы на контрольные вопросы:

1. Для чего используется JSON?

JSON – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

2. Какие типы значений используются в JSON?

Набор пар ключ: значение. Упорядоченный набор значений.

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?

Формат обмена данными JSON5 (JSON5) — это надмножество JSON, целью которого является смягчение некоторых ограничений JSON путем расширения его синтаксиса для включения некоторых продуктов из ECMAScript 5.1. Эта библиотека JavaScript является официальной эталонной реализацией библиотек синтаксического анализа и сериализации JSON5. Краткое описание возможностей. Следующие функции ECMAScript 5.1, которые не поддерживаются в JSON, были расширены до JSON5. Объекты. Ключи объекта могут быть идентификатором ECMAScript 5.1

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

Реализация Python формата данных JSON5. JSON5 расширяет формат обмена данными JSON, чтобы сделать его более удобным для использования в качестве языка конфигурации: Комментарии в стиле JavaScript (как однострочные, так и многострочные) разрешены. Ключи объектов могут быть без кавычек, если они являются допустимыми идентификаторами ECMAScript. Объекты и массивы могут заканчиваться запятыми. Строки могут заключаться в одинарные кавычки, допускаются много строчные строковые литералы. Есть еще несколько более мелких расширений JSON; см. полную информацию на странице выше. Этот проект реализует реализацию чтения и записи для Python; где воз можно, он отражает стандартный пакет Python JSON API для простоты использования. Есть одно заметное отличие от JSON api: методы load () и load () поддерживают опциональную проверку (и отклонение) повторяющихся ключей объекта; pass allow_duplicate_keys = False для этого (по умолчанию разрешены дубликаты). Это ранний выпуск. Это было достаточно хорошо протестировано, но это МЕДЛЕННО. Он может быть в 1000-6000 раз медленнее, чем модуль JSON, оптимизированный для C, и в 200 раз (или более) медленнее, чем модуль JSON на чистом Python

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

json.dump() # конвертировать python объект в json и записать в файл
json.dumps() # тоже самое, но в строку.

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

Dumps записывает в строку, а dump в файл.

8. Какие средства предоставляет Python для десериализации данных 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 кириллицу?

import codecs json.load(codecs.open('sample.json', 'r', 'utf-8-sig'))

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

Схема JSON — это словарь, который позволяет аннотировать и проверять

документы JSON.

Преимущества:

• Описывает ваш существующий формат (ы) данных.

• Предоставляет понятную документацию, читаемую человеком и

машиной.

• Проверяет данные, которые полезны для:

• Автоматизированное тестирование.

• Обеспечение качества предоставленных клиентом данных

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