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

Кафедра

инфокоммуникаций

Институт цифрового

развития

ОТЧЁТ

по лабораторной работе №2.16

Дисциплина: «Основы программной инженерии»

Тема: «Работа с данными формата JSON в языке Python»

Выполнил: студент
2 курса группы
Пиж-б-о-21-1
Рязанцев Матвей
Денисович

Ставрополь 2023

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

1. Создайте проект PyCharm в папке репозитория.

Рисунок 1 – Была создана папка PyCharm

2. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

```
>>> list
+------+--------------------------------+----------------------+----------+
|  №   |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | ebrfr                          | brvw                 |   221122 |
|    2 | vrw                            | wrvc                 |       11 |
|    3 | wegef                          | qgfac                |     2022 |
+------+--------------------------------+----------------------+----------+
>>> save data.json
```

Рисунок 3 – Результат работы программы

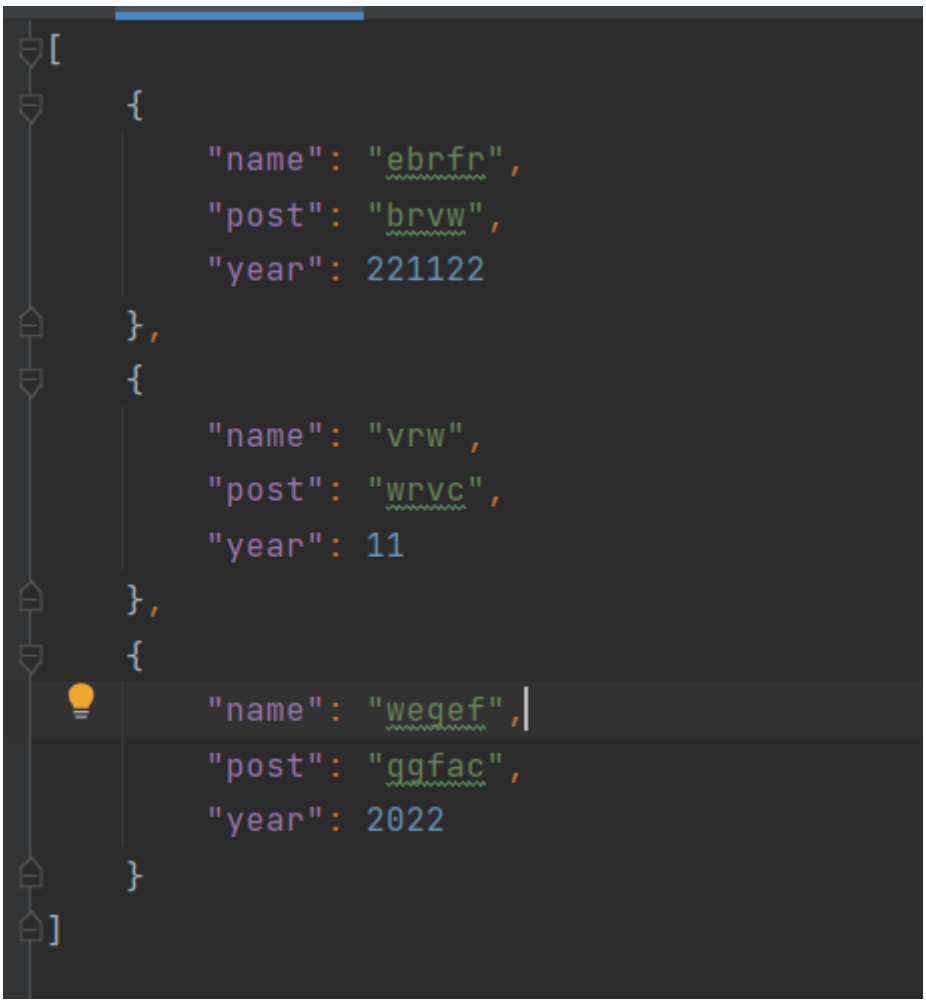

Рисунок 4 – Содержимое файла jfile.json

3. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

Код программы:

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

import json
import sys
from datetime import date


def add(pep):
     # Запросить данные о работнике.
    name = input("name faname? ")
    num = int(input("number? "))
    br = int(input("burftday? "))

        # Создать словарь.
    chel = {
            'name': name,
            'num': num,
            'br': br,
            }
```

```
            # Добавить словарь в список.
    pep.append(chel)
            # Отсортировать список в случае необходимости.
    if len(pep) > 1:
        pep.sort(key=lambda item: item.get('br',''))
    return pep

def li(pep):
     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
     print(line)
     print(
          '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
              "№",
              "F.I.O.",
              "NUMBER",
              "BRDAY"
          )
     )
     print(line)
     for idx, chel in enumerate(pep, 1):
        print(
             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                 idx,
                 chel.get('name', ''),
                 chel.get('num', ''),
                 chel.get('br', 0)
             )
        )
        print(line)


def sel(pep):
     # Получить требуемый стаж.
     zapros = int(input("zapros po numeru  "))

     # Инициализировать счетчик.
     count = 0
     # Проверить сведения работников из списка.
     for chel in pep:
        if chel.get('num') == zapros:
            count += 1
            print(
                 '{:>4}: {}'.format(count, chel.get('name', ''))
            )

            # Если счетчик равен 0, то работники не найдены.
     if count == 0:
        print("cheela s takim nomerom net")


def save_workers(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
```

```
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
# Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


if __name__ == '__main__':
    # Список работников.
    pep = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            pep = add(pep)

        elif command == 'list':
            li(pep)
        elif command == 'select':
            sel(pep)
        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, pep)

        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.
            pep = load_workers(file_name)
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - add chel;")
            print("list - show list of pep;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
D:\2kurs\!22kurs\opi\opi_lab2-16\Scripts\python.exe D:\2kurs\!22kurs\opi_lab2-16\idz.py
>>> load date1.json
>>> list
+------+--------------------------------+----------------------+----------+
|  №   |            F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+----------+
|    1 | rtngr                          | 22                   |   223452 |
+------+--------------------------------+----------------------+----------+
>>>
```

Рисунок 5 – Результат работы программы

Рисунок 6 – Содержимое файла indjs.json

Задание повышенной сложности

Код программы:

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

import json
import sys
from datetime import date
import jsonschema


def add(pep):
     # Запросить данные о работнике.
    name = input("name faname? ")
    num = int(input("number? "))
    br = int(input("burftday? "))

        # Создать словарь.
    chel = {
            'name': name,
            'num': num,
            'br': br,
            }

            # Добавить словарь в список.
```

```
    pep.append(chel)
            # Отсортировать список в случае необходимости.
    if len(pep) > 1:
        pep.sort(key=lambda item: item.get('br',''))
    return pep

def li(pep):
     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
     print(line)
     print(
          '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
              "№",
              "F.I.O.",
              "NUMBER",
              "BRDAY"
          )
     )
     print(line)
     for idx, chel in enumerate(pep, 1):
        print(
             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                 idx,
                 chel.get('name', ''),
                 chel.get('num', ''),
                 chel.get('br', 0)
             )
        )
        print(line)


def sel(pep):
     # Получить требуемый стаж.
     zapros = int(input("zapros po numeru  "))

     # Инициализировать счетчик.
     count = 0
     # Проверить сведения работников из списка.
     for chel in pep:
        if chel.get('num') == zapros:
            count += 1
            print(
                 '{:>4}: {}'.format(count, chel.get('name', ''))
            )

            # Если счетчик равен 0, то работники не найдены.
     if count == 0:
        print("cheela s takim nomerom net")


def save_workers(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)
```

```
def load_workers(file_name):
    schema = {
      "type": "array",
      "items": [
        {
          "type": "object",
          "properties": {
            "name": {
              "type": "string"
            },
            "num": {
              "type": "integer"
            },
            "br": {
              "type": "integer"
            }
          },
          "required": [
            "name",
            "num",
            "br"
          ]
        }
      ]
    }
    with open(file_name, "r", encoding="utf-8") as fin:
        loadfile = json.load(fin)
        validator = jsonschema.Draft7Validator(schema)
        try:
            if not validator.validate(loadfile):
                print("валидация успешна")
        except jsonschema.exceptions.ValidationError:
            print("ошибка валидации", file=sys.stderr)
            exit()
    return loadfile


if __name__ == '__main__':
    # Список работников.
    pep = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            pep = add(pep)

        elif command == 'list':
            li(pep)
        elif command == 'select':
            sel(pep)
        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, pep)

        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.
            pep = load_workers(file_name)
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - add chel;")
            print("list - show list of pep;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
D:\2kurs\!22kurs\opi\opi_lab2-16\Scripts\python.exe D:\2kurs\!22kurs
>>> load datapovs.json
валидация успешна
>>> add
name faname? mat
number? 2
burftday? 22
>>> add
name faname? m
number? 3
burftday? 11
>>> save datapovs.json
>>> load datapovs.json
валидация успешна
```

Рисунок 7 – Результат работы программы

Контр. вопросы и ответы на них:

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как JAYsən)

– текстовый формат обмена данными, основанный на JavaScript. Как имногие

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 от формата данных JSON?

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 многострочные 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 возможно, 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 язык 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 функций json.dump() и json.dumps()?

Dumps записывает в строку, a dump в файл.

8. Какие средства предоставляет язык Python для десериализации 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?

```
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.