МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Работа с данными формата JSON в языке Python»

Отчет по лабораторной работе № 2.16 по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-21-1

Пуценко И. А. « » 2022г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____ (подпись)

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

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.
8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.
9. Зафиксируйте сделанные изменения в репозитории.
10. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.
11. Зафиксируйте сделанные изменения в репозитории.
12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
13. Выполните слияние ветки для разработки с веткой master/main.
14. Отправьте сделанные изменения на сервер GitHub.
15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Проработка примеров:

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

import sys
import json
from datetime import date


def get_worker() -> dict:
    """
    Запросить данные о работнике.
    :return dict:
    """
    name = input("Фамилия и инициалы >> ")
    post = input("Должность >> ")
    year = int(input("Год поступления >> "))

    return {
        'name': name,
        'post': post,
        'year': year
    }


def display_workers(staff):
    """
    Отобразить список работников.
    :param staff:
    :return:
    """
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
                "№",
                "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 сотрудников пуст.")
```

```python
def select_workers(staff, period: int) -> list:
    """
    Выбрать работников с заданным стажем
    :param staff:
    :param period:
    :return result:
    """
    today = date.today()

    result = list()

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

    return result


def save_workers(file_name: str, staff):
    """
    Сохранить всех работников в файл JSON.
    :param file_name:
    :param staff:
    :return:
    """
    with open(file_name, 'w', encoding='utf-8') as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_workers(file_name: str) -> dict:
    """
    Загрузить всех работников из файла JSON.
    :param file_name:
    :return dict:
    """
    with open(file_name, 'r', encoding='utf-8') as fin:
        return json.load(fin)


def main():
    """
        Главная функция программы
        """
    # Список работников.
    workers = list()

    while True:
        # Запросить команду из терминала
        command = input(">>> ").lower()

        # Выполнить действие в соответствии с командой.
        if command == "exit":
            break

        elif command == "add":
            worker = get_worker()

            workers.append(worker)

            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 "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]

            save_workers(file_name, workers)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]

            workers = load_workers(file_name)

        elif command == "help":
            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 - завершиьт работу с программой.")

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


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

Листинг 1 – Код примера работы с JSON-файламы

Самостоятельные задания

Задание 1.

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.*

Прописать валидацию данных с использованием библиотеки jsonschema.

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

import json
import os
```

```
import jsonschema
from jsonschema import validate


def load_data():
    data = []
    if os.path.exists(data_file):
        with open(data_file, "r") as file:
            data = json.load(file)
            validate(data, schema)
    return data


def save_data(data):
    validate(data, schema)
    with open(data_file, "w") as file:
        json.dump(data, file, indent=4)


def exit_to_program():
    print('всего доброго')
    save_data(lst_planes)
    return exit(1)


def help_program():
    print("add - добавление рейса\n"
          "help - помощь по командам\n"
          "select \"пункт назначения\" - вывод самолетов летящих в п.н.\n"
          "display_plane - вывод всех самолетов\n"
          "exit - выход из программы")


def add_program(planes):
    plane = dict()
    plane["destination"] = input("Пункт назначения:\n")
    plane["flight_number"] = int(input("Номер рейса:\n"))
    plane["type_plane"] = input("Тип самолета\n")
    planes.append(plane)
    planes.sort(key=lambda key_plane: key_plane.get("flight_number"))
    return planes


def select_program(planes):
    lst = list(map(lambda x: x.get("destination"), planes))
    point = input('выберите нужное вам место\n')
    print("результаты поиска")
    if point in lst:
        print('рейсы в эту точку')
        for i in planes:
            if point == i["destination"]:
                print(f"{i['flight_number']}........{i['type_plane']}")
    else:
        print("рейсов не найдено")


def error():
    print('неверная комманда')


def display_plane(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
```

```
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Направление",
                "Тип самолета",
                "рейс"
            )
        )
        print(line)

        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('destination', ''),
                    worker.get('type_plane', ''),
                    worker.get('flight_number', 0)
                )
            )
            print(line)

    else:
        print("рейсов не найдено")


def menu(lst_plane):
    command = input('введите команду("help" - руководство по
командам)\n>>>').lower()
    if command == 'exit':
        exit_to_program()
    elif command == 'help':
        help_program()
    elif command == 'add':
        lst_plane = add_program(lst_plane)
    elif command == 'select':
        select_program(lst_plane)
    elif command == 'display_plane':
        display_plane(lst_plane)
    else:
        error()


if __name__ == '__main__':
    data_file = input("введите расположение файла: ")

    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "destination": {"type": "string"},
                "flight_number": {"type": "integer"},
                "type_plane": {"type": "string"}
            },
            "required": ["destination", "flight_number", "type_plane"]
        }
    }

    lst_planes = load_data()
```

```
    while True:
        menu(lst_planes)
```

Листинг 3 – Валидация данных с помощью jsonschema

```
введите расположение файла: data.json
Traceback (most recent call last):
  File "C:\Users\FonK\Desktop\python\OPI\labRabOPI_2.16\PyCharm\individual.py", line 132, in <module>
    lst_planes = load_data()
  File "C:\Users\FonK\Desktop\python\OPI\labRabOPI_2.16\PyCharm\individual.py", line 15, in load_data
    validate(data, schema)
  File "C:\Users\FonK\Desktop\python\OPI\labRabOPI_2.16\venv\lib\site-packages\jsonschema\validators.py", line 1121, in validate
    raise error
jsonschema.exceptions.ValidationError: 'qw' is not of type 'integer'

Failed validating 'type' in schema['items']['properties']['flight_number']:
    {'type': 'integer'}

On instance[1]['flight_number']:
    'qw'
```

Рисунок 1 – Исключение при ошибке валидации

Вопросы для защиты работы

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.

2. Какие типы значений используются в JSON?

   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 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?

Объекты

- *Объекты могут иметь одну запятую.*

Массивы

- *Массивы могут иметь одну запятую.*

Строки

- *Строки могут заключаться в одинарные кавычки.*
- *Строки могут охватывать несколько строк, экранируя символы новой строки.*
- *Строки могут включать в себя экранирование символов.*

Числа

- *Числа могут быть шестнадцатеричными.*
- *Числа могут иметь ведущую или последующую десятичную точку.*
- *Числа могут быть Infinity, -Infinity2 и NaN.*
- *Числа могут начинаться с явно определенного знака +.*

Комментарии

- *Допускаются однострочные и многострочные комментарии.*

Пробельные символы

- *Разрешены дополнительные пробельные символы.*

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

Библиотека JSON, позволяющая работать с данными dict, str в python для преобразования их в json-формат, а также в обратную сторону, для преобразования json-файлов в читаемые python форматы.

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

json.dump() и json.dumps()

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

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?

   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?

   Нужно использоавть аргумент encoding со значением utf-8 в функции open при открытии JSON-файла.

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? Приведите схему данных для примера 1.

    Схема данных – это перечисление полей и их типов, которые должны быть в JSON. При несоблюдении названия, количества или типа полей, выдаётся ошибка валидации данных.

    Схема данных для примера 1:

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