

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Анализ данных»

Выполнил:
Кожуховский Виктор Андреевич
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 Роман Александрович

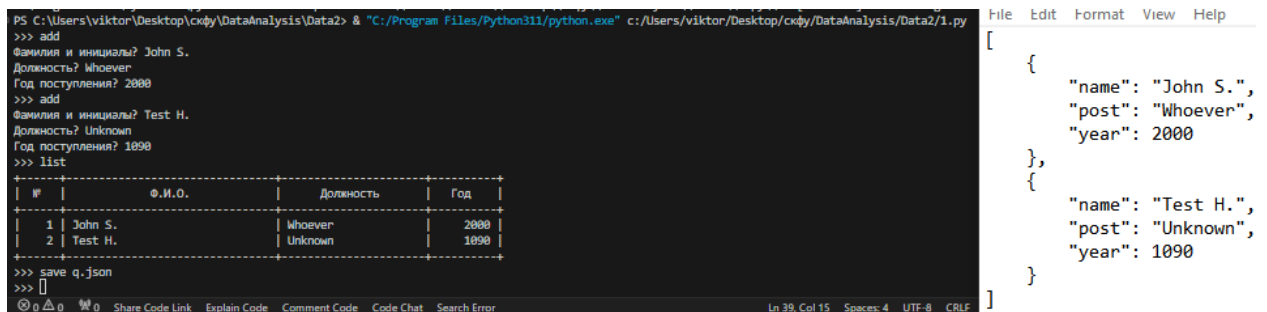
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Порядок выполнения работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.
3. Выполнил клонирование созданного репозитория.
4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow.
6. Создал проект в папке репозитория.
7. Проработал пример лабораторной работы.



```
PS C:\Users\viktor\Desktop\ckfy\DataAnalysis\Data2> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ckfy/DataAnalysis/Data2/1.py
>>> add
Фамилия и инициалы? John S.
Должность? Whoever
Год поступления? 2000
>>> add
Фамилия и инициалы? Test H.
Должность? Unknown
Год поступления? 1090
>>> list
+-----+-----+-----+-----+
| # | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | John S. | Whoever   | 2000 |
| 2 | Test H. | Unknown   | 1090 |
+-----+-----+-----+-----+
>>> save q.json
>>> 
```

```
[
  {
    "name": "John S.",
    "post": "Whoever",
    "year": 2000
  },
  {
    "name": "Test H.",
    "post": "Unknown",
    "year": 1090
  }
]
```

Рисунок 1. Выполнение примера 1

8. Выполнил индивидуальное задание и задание повышенной сложности.

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.


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

import json
import sys
from datetime import datetime
import jsonschema

person_schema = {
    "type": "object",
    "properties": {
        "name": {"type": "string"},
        "surname": {"type": "string"},
        "date_of_birth": {"type": "string", "format": "date"},
        "zodiac_sign": {"type": "string"}
    },
    "required": ["name", "surname", "date_of_birth", "zodiac_sign"]
}

def add_person(people):
    """
    Добавление нового человека в список.
    Список сортируется по знаку зодиака после добавления нового элемента.
    """
    name = input("Введите имя: ")
    surname = input("Введите фамилию: ")
    date_of_birth = datetime.strptime(
        input("Введите дату рождения (в формате ДД.ММ.ГГГГ через точку): "),
        "%d.%m.%Y"
    )
    zodiac_sign = input("Введите знак зодиака: ")

    person = {
        "name": name,
        "surname": surname,
        "date_of_birth": date_of_birth,
        "zodiac_sign": zodiac_sign
    }

    people.append(person)
    people.sort(key=lambda item: item.get('zodiac_sign', ''))

def list_people(people):
    """
    Вывод таблицы людей.
    """
    line = "{:4} | {:20} | {:20} | {:15} | {:12} |".format(
        "№", "Имя", "Фамилия", "Знак Зодиака", "Дата рождения"
    )
    print(line)
    print(
        "{:4} | {:20} | {:20} | {:15} | {:12} |".format(
            "№", "Имя", "Фамилия", "Знак Зодиака", "Дата рождения"
        )
    )
    print(line)

    for idx, person in enumerate(people, 1):
        birth_date_str = person.get('date_of_birth').strftime('%d.%m.%Y')
        print(
            "{:4} | {:20} | {:20} | {:15} | {:12} |".format(
                idx,
                person.get('name', ''),
                person.get('surname', ''),
                person.get('zodiac_sign', ''),
                birth_date_str
            )
        )
    print(line)

def select_people(people, month):
    """
    Вывести список людей, родившихся в заданном месяце.
    """
    count = 0
    for person in people:
        if person.get('date_of_birth').month == month:
            count += 1
            print("{:4} | {:20} | {:20} | {:15} | {:12} |".format(
                count, person.get('name', ''), person.get('surname', ''),
                person.get('zodiac_sign', ''),
                person.get('date_of_birth').strftime('%d.%m.%Y')
            ))
    if count == 0:
        print("Никто не родился в указанный месяц, не найдено.")

def show_help():
    """
    Вывести справку по командам программы.
    """
    print("Список команд:\n")
    print("add - добавить человека;\n")
    print("list - вывести список людей;\n")
    print("select месяц - вывод на экран информации о людях,\n")
    print("родившихся в указанный месяц (цифра)\n")
    print("help - отобразить справку;\n")
    print("load - загрузить данные из файла;\n")
    print("save - сохранить данные в файл;\n")
    print("exit - завершить работу с программой.")

def save_people(file_name, staff):
    """
    Сохранить всех людей в файл JSON.
    """
    staff_formatted = [{"person": person.get('date_of_birth'): person.get(
        'date_of_birth').strftime('%d.%m.%Y')} for person in staff]
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выгрузить сериализованные данные в формат JSON.
        json.dump(staff_formatted, fout, ensure_ascii=False, indent=4)

def load_people(file_name):
    """
    Загрузить всех людей из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        staff_loaded = json.load(fin)
        result_people = []
        cnt = 0
        for person in staff_loaded:
            cnt += 1
            if validate_person(person, person_schema):
                try:
                    person['date_of_birth'] = datetime.strptime(
                        person['date_of_birth'], '%d.%m.%Y')
                    result_people.append(person)
                except:
                    print("Ошибка при разборе даты в записи, пропуск записи (cnt).")
            else:
                print("Неверные данные человека, пропуск записи.")
        return result_people

def validate_person(person_data, schema):
    """
    Проверка валидности данных.
    """
    try:
        jsonschema.validate(person_data, schema)
        return True
    except jsonschema.exceptions.ValidationError as e:
        print(f"Данные человека не соответствуют схеме: {e}")
        return False

def main():
    """
    Основная программа.
    """
    people = []

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

        if command == 'exit':
            break
        elif command == 'add':
            add_person(people)
        elif command == 'list':
            list_people(people)
        elif command.startswith('select '):
            month = int(command.split(' ')[1].split('-')[0])
            select_people(people, month)
        elif command.startswith('save '):
            file_name = command.split(' ')[1] + ".json"
            save_people(file_name, people)
        elif command.startswith('load '):
            file_name = command.split(' ')[1] + ".json"
            people = load_people(file_name)
        else:
            print("Неизвестная команда (command)", file=sys.stderr)

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

Рисунок 3. Код решения задания повышенной сложности и его выполнение

- 9. Зафиксировал сделанные изменения в репозитории.
- 10. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.
- 11. Выполнил слияние ветки для разработки с веткой master/main.
- 12. Отправил сделанные изменения на сервер GitHub.

Контрольные вопросы:

- 1. Для чего используется JSON?
JSON используется для обмена данными между приложениями. Это легкий формат обмена данными, который удобен для чтения и записи как людьми, так и компьютерами.
- 2. Какие типы значений используются в JSON?

В JSON используются следующие типы значений: строки (в двойных кавычках), числа, логические значения (true/false), массивы (списки значений), объекты (связанные пары ключ-значение), null.

3. Как организована работа со сложными данными в JSON?

Для работы со сложными данными в JSON используется структура объектов, массивов и вложенных объектов, позволяющая организовать информацию иерархически.

4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?

Формат данных JSON5 - это расширение JSON, поддерживающее комментарии, необязательные запятые в конце элементов массива и объекта, поддержку некоторых дополнительных типов данных, таких как даты. Основное отличие от JSON - это более гибкий и читаемый синтаксис.

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

Для работы с данными в формате JSON5 в Python можно использовать библиотеку `json5` или другие сторонние библиотеки, которые поддерживают этот формат.

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

В Python для сериализации данных в формат JSON используются встроенная библиотека `json` и метод `json.dumps()`.

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

`json.dump()` используется для записи данных в поток (например, файл), а `json.dumps()` используется для преобразования данных в строку.

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

Для десериализации данных из формата JSON в Python используется метод `json.loads()` и библиотека `json`.

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

Для работы с данными формата JSON, содержащими кириллицу, необходимо использовать кодировку UTF-8 при чтении и записи данных с помощью библиотеки json в Python.

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

Схема данных представляет собой описание ожидаемой структуры JSON, включая типы данных, ограничения и правила валидации, которые должны соответствовать данным.

Схема данных для примера 1:

```
{  
  "type": "object",  
  "properties": {  
    "name": {"type": "string"},  
    "post": {"type": "string"},  
    "year": {"type": "integer"}  
  },  
  "required": ["name", "post", "year"]  
}
```