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

Кучеренко С. Ю. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2023

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 IT 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. Проработать примеры лабораторной работы. Создайте для них отдельные модули языка. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

Пример 1. Для примера 1 лабораторной работы 2.8 добавьте возможность сохранения списка в файл формата JSON и чтения данных из файла JSON.

```
#!/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):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
if staff:
        # Заголовок таблицы.          line =
'+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
'-' * 8
        )
print(line)
print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
)
print(line)
        # Вывести данные о всех сотрудниках.
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):
    """
    Выбрать работников с заданным стажем.
"""


    # Получить текущую дату.
    today = date.today()
    # Сформировать список работников.
    result = []     for employee in staff:         if today.year -
employee.get('year', today.year) >= period:
            result.append(employee)
    # Возвратить список выбранных работников.
    return result
```

8. Зафиксируйте сделанные изменения в репозитории.

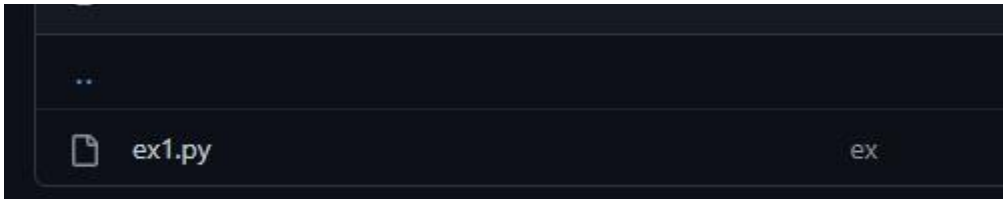

Рисунок 1 – Фиксирование изменений в репозитории

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

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.

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

import json
import sys


def get_way():
    """
    Запросить данные о маршруте.
    """
    start = input('Название начального маршрута: ')
    finish = input('Название конечного маршрута: ')
    num = int(input('Номер маршрута: '))

    # Вернуть словарь.
    return {
        'start': start,
        'finish': finish,
        'num': num
    }


def display_way(numbers):
    """
    Отобразить список маршрутов.
    """
    if numbers:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 15
        )
```

```
        print(line)
        print(
            '| {:^4} | {:^30} | {:^30} | {:^15} |'.format(
                "No",
                "Название начального маршрута",
                "Название конечного маршрута",
                "Номер маршрута"
            )
        )
        print(line)

        # Вывести данные о всех маршрутах.
        for idx, way in enumerate(numbers, 1):
            print(
                '| {:>4} | {:<30} | {:<30} | {:>15} |'.format(
                    idx,
                    way.get('start', ''),
                    way.get('finish', ''),
                    way.get('num', 0)
                )
            )
        print(line)

    else:
        print("Список пуст.")


def find_way(numbers, nw):
    """
    Выбрать маршрут с данным номером.
    """
    # Список маршрутов
    result = []
    for h in numbers:
        if nw in str(h.values()):
            result.append(h)

    # Проверка на наличие записей
    if len(result) == 0:
        return None

    # Возвратить список выбранных маршрутов.
    return result


def save_ways(file_name, routes):
    """
    Сохранить номера всех маршрутов 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(routes, fout, ensure_ascii=False, indent=4)


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

```python
def main():
    """
    Главная функция программы.
    """
    # Маршруты
    ways = []

    # 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':
            # Запросить данные о маршруте.
            way = get_way()

            # Добавить словарь в список.
            ways.append(way)
            # Отсортировать список в случае необходимости.
            if len(ways) > 1:
                ways.sort(key=lambda item: item.get('num', 0))

        elif command == 'list':
            # Отобразить все маршруты.
            display_way(ways)

        elif command == 'find':
            f = input('Введите номер маршрута: ')
            selected = find_way(ways, f)
            display_way(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_ways(file_name, ways)

        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.
            ways = load_ways(file_name)

        elif command == 'help':
            # Вывести справку.
            print("Список команд:\n")
            print("add - добавить маршрут;")
            print("list - вывести список маршрутов;")
            print("find - вывод информации о маршруте;")
            print("save - сохранить данные в файл;")
            print("load - загрузить данные из файла;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        else:
```

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


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

```
/Users/svetik/Desktop/OPI/OPI_LR_2.16/PyCharm/venv/bin/python /Users/svetik/Desktop/OPI/OPI_LR_2
>>> add
Название начального маршрута: lalal
Название конечного маршрута: pupupu
Номер маршрута: 55
>>> add
Название начального маршрута: auauau
Название конечного маршрута: totoot
Номер маршрута: 2
>>> list
+------+--------------------------------+--------------------------------+-----------------+
|  No  |  Название начального маршрута  |  Название конечного маршрута   | Номер маршрута  |
+------+--------------------------------+--------------------------------+-----------------+
|    1 | auauau                         | totoot                         |               2 |
|    2 | lalal                          | pupupu                         |              55 |
+------+--------------------------------+--------------------------------+-----------------+
>>> save data_ind
>>> load data_ind
>>> list
+------+--------------------------------+--------------------------------+-----------------+
|  No  |  Название начального маршрута  |  Название конечного маршрута   | Номер маршрута  |
+------+--------------------------------+--------------------------------+-----------------+
|    1 | auauau                         | totoot                         |               2 |
|    2 | lalal                          | pupupu                         |              55 |
+------+--------------------------------+--------------------------------+-----------------+
>>>
```

Рисунок 2 – Результат выполнения программы

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-sch ema.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 json
import sys
import os
import jsonschema
from jsonschema import validate


def get_way():
    """
    Запросить данные о маршруте.
    """
    start = input('Название начального маршрута: ')
    finish = input('Название конечного маршрута: ')
    num = int(input('Номер маршрута: '))

    # Вернуть словарь.
    return {
        'start': start,
        'finish': finish,
        'num': num,
    }


def display_way(numbers):
    """
    Отобразить список маршрутов.
    """
    if numbers:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^30} | {:^15} |'.format(
                "No",
                "Название начального маршрута",
                "Название конечного маршрута",
                "Номер маршрута"
            )
        )
        print(line)

        # Вывести данные о всех маршрутах.
        for idx, way in enumerate(numbers, 1):
            print(
                '| {:>4} | {:<30} | {:<30} | {:>15} |'.format(
                    idx,
                    way.get('start', ''),
                    way.get('finish', ''),
                    way.get('num', 0)
                )
            )
```

```
        print(line)

    else:
        print("Список пуст.")


def find_way(numbers, nw):
    """
    Выбрать маршрут с данным номером.
    """
    # Список маршрутов
    result = []
    for h in numbers:
        if nw in str(h.values()):
            result.append(h)
    # Возвратить список выбранных маршрутов.
    return result


def save_ways(file_name, routes):
    """
    Сохранить номера всех маршрутов 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(routes, fout, ensure_ascii=False, indent=4)


def load_ways(file_name, load):
    os.chdir("/Users/svetik/Desktop/OPI/OPI_LR_2.16")
    """
    Загрузить все маршруты из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)
    print("Файл загружен")
    validate(file, load)
    return file


def validate(file, schema):
    validator = jsonschema.Draft7Validator(schema)
    try:
        if not validator.validate(file):
            print("Нет ошибок валидации")
    except jsonschema.exceptions.ValidationError:
        print("Ошибка валидации", file=sys.stderr)
        exit(1)


def main():
    """
    Главная функция программы.
    """
    os.chdir("/Users/svetik/Desktop/OPI/OPI_LR_2.16")
    with open('check.json', 'r') as check:
        first_load = json.load(check)
    # Маршруты
    ways = []

    # 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':
            # Запросить данные о маршруте.
            way = get_way()

            # Добавить словарь в список.
            ways.append(way)
            # Отсортировать список в случае необходимости.
            if len(ways) > 1:
                ways.sort(key=lambda item: item.get('num', 0))

        elif command == 'list':
            # Отобразить все маршруты.
            display_way(ways)

        elif command == 'find':
            f = command.split('Введите номер маршрута: ', maxsplit=1)
            period = int(f[1])

            selected = find_way(ways, period)
            display_way(selected)

        elif command.startswith("save "):
            # Разбить команду на части для выделения имени файла.
            f = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = f[1]
            # Сохранить данные в файл с заданным именем.
            save_ways(file_name, ways)

        elif command.startswith("load "):
            # Разбить команду на части для выделения имени файла.
            f = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = f[1]
            # Сохранить данные в файл с заданным именем.
            ways = load_ways(file_name, first_load)

        elif command == 'help':
            # Вывести справку.
            print("Список команд:\n")
            print("add - добавить маршрут;")
            print("list - вывести список маршрутов;")
            print("find - вывод информации о маршруте;")
            print("save - сохранить данные в файл;")
            print("load - загрузить данные из файла;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

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


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

10. Зафиксируйте сделанные изменения в репозитории.

Вопросы для защиты работы:

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как /ˈdʒeɪsən/ JAY-sən) - текстовый формат обмена данными, основанный на JavaScript. 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).

2. Какие типы значений используются в JSON?

Если быть точным, то им нужно быть одним из шести типов данных: строкой, числом, объектом, массивом, булевым значением или null.

Как было показано ранее JSON-текст представляет собой (в закодированном виде) одну из двух структур:

Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимость не регулируется стандартом, это остаётся на усмотрение программного обеспечения. Как правило, регистр учитывается программами — имена с буквами в разных регистрах считаются разными, значением — любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.

Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

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 escapeпоследовательностей, 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?

Вложенные объекты

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. Фигурные скобки везде используются для

формирования вложенного JSON объекта с ассоциированными именами пользователей и данными локаций для каждого из них. Как и с любым другим значением, используя объекты, двоеточие используется для разделения элементов.

Вложенные массивы

Данные также могут быть вложены в формате JSON, используя JavaScript массивы, которые передаются как значения. JavaScript использует квадратные скобки [ ] для формирования массива. Массивы по своей сути — это упорядоченные коллекции и могут включать в себя значения совершенно разных типов данных. Мы можем использовать массив при работе с большим количеством данных, которые могут быть легко сгруппированны вместе, как например, если есть несколько разных сайтов и профайлов в социальных сетях ассоциированных с одним пользователем.

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

Формат обмена данными JSON5 – это расширенная JSON-версия, которая призвана смягчить некоторые ограничения JSON, расширив его синтаксис и включив в него некоторые функции из ECMAScript 5.1.

Некоторые нововведения:

− Поддерживаются как однострочные //, так и многострочные /* */ комментарии.

− Записи и списки могут иметь запятую после последнего элемента (удобно при копировании элементов).

− Ключи записей могут быть без кавычек, если они являются валидными идентификаторами ECMAScript 5.

− Строки могут заключаться как в одинарные, так и в двойные кавычки.

− Числа могут быть в шестнадцатеричном виде, начинаться или заканчиваться десятичной точкой, включать Infinity, -Infinity, NaN и -NaN, начинаться со знака +.

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

Упаковка объектов в байтовую последовательность называется сериализацией. А распаковка байтов в объекты языка программирования, приведение последовательности назад к типам и структурам, — десериализацией.

Dumps позволяет создать JSON-строку из переданного в нее объекта. Loads — преобразовать строку назад в объекты языка.

Dump и load используют, чтобы сохранить результат в файл или воссоздать объект. Работают они схожим образом, но требуют передачи специального объекта для работы с файлом — filehandler.

Пользовательские классы не относятся к JSON-сериализуемым. Это значит, что просто применить к ним функции dumps, loads или dump и load не получится.

Чтобы сериализовать пользовательский объект в JSON-структуру данных, нужен аргумент default. Указывайте вызываемый объект, то есть функцию или статический метод.

Чтобы получить аргументы класса с их значениями, нужна встроенная функция __dict__, потому что любой класс — это словарь со ссылками на значения по ключу.

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

Сериализация данных в формат JSON:

json.dump() # конвертировать python объект в json и записать в файл

json.dumps() # тоже самое, но в строку

Обе эти функции принимают следующие необязательные аргументы:

Если skipkeys = True , то ключи словаря не базового типа (str , int , float, bool , None) будут проигнорированы, вместо того, чтобы вызывать исключение TypeError .

Если ensure_ascii = True , все не-ASCII символы в выводе будут экранированы последовательностями \uXXXX , и результатом будет строка, содержащая только ASCII символы. Если ensure_ascii = False , строки запишутся как есть.

Если check_circular = False , то проверка циклических ссылок будет пропущена, а такие ссылки будут вызывать OverflowError .

Если allow_nan = False , при попытке сериализовать значение с запятой, выходящее за допустимые пределы, будет вызываться ValueError (nan, inf, inf) в строгом соответствии со спецификацией JSON, вместо того, чтобы использовать эквиваленты из JavaScript (NaN, Infinity, -Infinity).

Если indent является неотрицательным числом, то массивы и объекты в JSON будут выводиться с этим уровнем отступа. Если уровень отступа 0, отрицательный или "", то вместо этого будут просто использоваться новые строки. Значение по умолчанию None отражает наиболее компактное представление. Если indent - строка, то она и будет использоваться в качестве отступа.

Если sort_keys = True , то ключи выводимого словаря будут отсортированы.

7. В чем отличие функций json.dump() и json.dumps()? json.dumps() конвертирует python объект в json и записывает его в строку

вместо записи в файл.

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

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) Обе эти функции принимают следующие аргументы:

object_hook - опциональная функция, которая применяется к результату декодирования объекта (dict). Использоваться будет значение, возвращаемое этой функцией, а не полученный словарь.

object_pairs_hook - опциональная функция, которая применяется к результату декодирования объекта с определённой последовательностью пар ключ/значение. Будет использован результат, возвращаемый функцией, вместо исходного словаря. Если задан так же object_hook , то приоритет отдаётся object_pairs_hook .

parse_float , если определён, будет вызван для каждого значения JSON с плавающей точкой. По умолчанию, это эквивалентно float(num_str) .

parse_int , если определён, будет вызван для строки JSON с числовым значением. По умолчанию эквивалентно int(num_str) .

parse_constant , если определён, будет вызван для следующих строк: "-Infinity", "Infinity", "NaN". Может быть использовано для возбуждения исключений при обнаружении ошибочных чисел JSON.

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

Использование кодировки UTF-8 или ensure_ascii=False

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

Схема данных представляет собой код, который используется для валидации данных в формате JSON. Она описывает ваш существующий формат (ы) данных, предоставляет понятную документацию для чтения человеком и машиной, проверяет данные, которые полезны для: автоматизированного тестирования, обеспечение качества предоставляемых клиентом данных.