

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 №4
дисциплины «Основы программной инженерии»

Выполнил:
Мелтонян Одиссей
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:
Кандидат технических наук, доцент
кафедры инфокоммуникаций
Воронкин Р. А

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г

Тема: Работа с данными формата JSON в языке Python

Цель работы: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python

Ход работы:

1. Изучен теоретический материал работы.
2. Создан общедоступный репозиторий на GitHub, в котором была использована лицензия MIT и язык программирования Python.

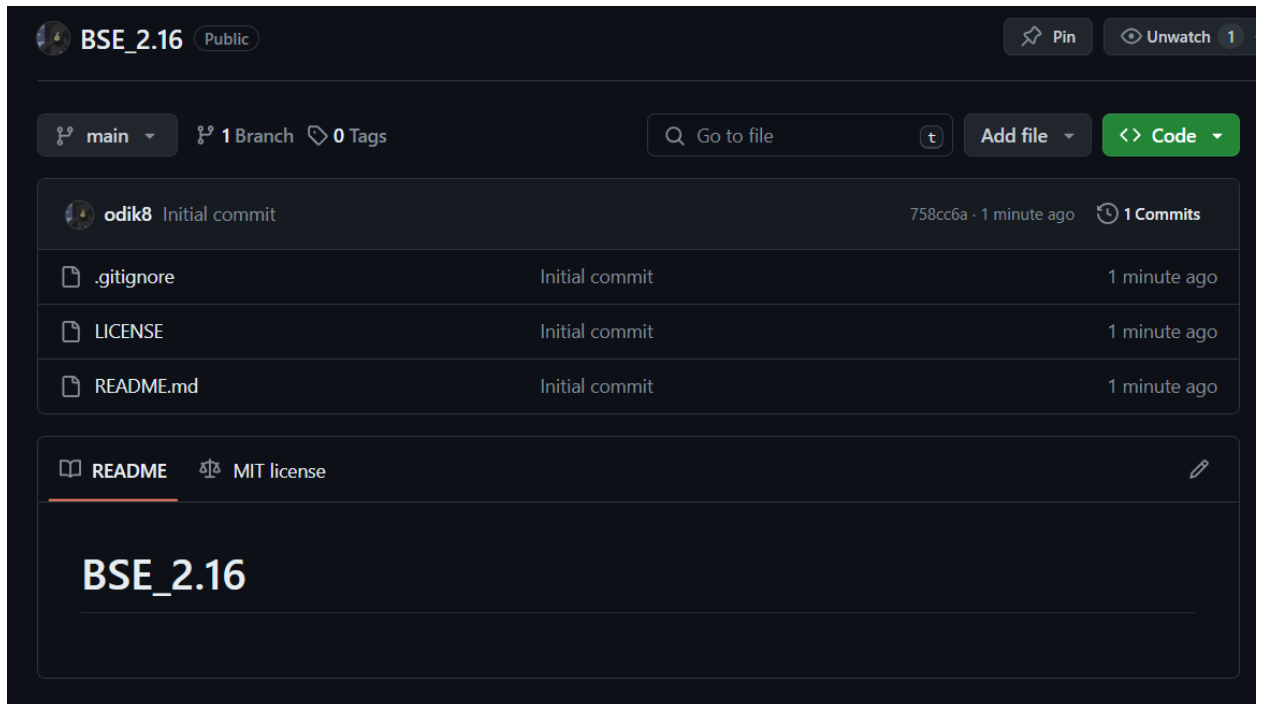


Рисунок 1 – Созданный репозиторий

3. Выполнено клонирование созданного репозитория.

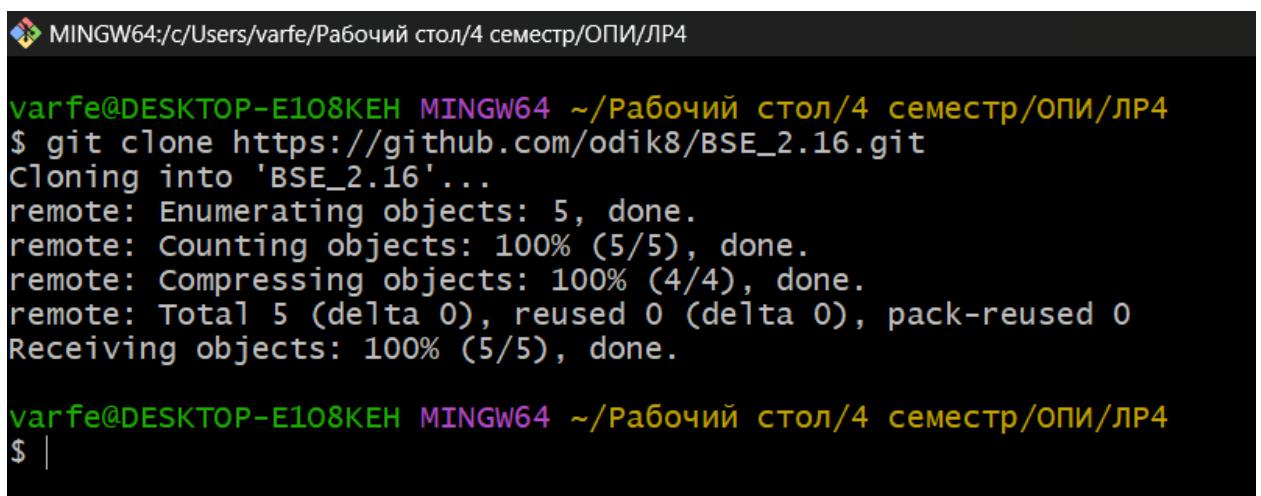


Рисунок 2 – Клонирование репозитория

4. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организован репозиторий в соответствие с моделью ветвления git-flow.

```
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР4/bse_2.16 (main)
$ git flow init
warning: ignoring broken ref refs/heads/desktop.ini

Which branch should be used for bringing forth production releases?
warning: ignoring broken ref refs/heads/desktop.ini
- main
Branch name for production releases: [main]
warning: ignoring broken ref refs/heads/desktop.ini
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/varfe/Рабочий стол/4 семестр/ОПИ/ЛР4/bse_2.16/.git/hooks]

varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР4/bse_2.16 (develop)
$ |
```

Рисунок 3 – Инициализация git flow

6. Для задания из лабораторной работы 2.8 дополнительно реализованы функции сохранения и чтение данных из файла формате JSON.

```
def save_to_json(file_name, list_of_people):
    """
    Сохранить всех в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        print(list_of_people)
        json.dump(list_of_people, fout, ensure_ascii=False, indent=4)
    print("Данные успешно сохранены в файл", file_name)

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

Рисунок 4 – Новые функции

Код всей программы:

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

from datetime import datetime
import json

def get_birthdate():
    while True:
        try:
            date_str = input("Введите дату рождения в формате ДД.ММ.ГГГГ: ")
            birthdate = datetime.strptime(date_str, "%d.%m.%Y").date()
            return birthdate
        except ValueError:
            print("Ошибка Неправильный формат даты. Попробуйте снова.")

def add_person(list_of_people):
    print("\nДобавление нового человека:")
    last_name = input("Введите фамилию: ")
    first_name = input("Введите имя: ")
    phone_number = input("Введите номер телефона: ")
    birthdate = get_birthdate()

    person = {
        'фамилия': last_name,
        'имя': first_name,
        'номер телефона': phone_number,
        'дата рождения': str(birthdate),
    }

    list_of_people.append(person)
    list_of_people.sort(key=lambda x: x['дата рождения'])
    print("Человек добавлен\n")

def find_person_by_phone(people, phone):
    for person in people:
        match person['номер телефона']:
            case phone:
                return person
    return None

def print_person_info(list_of_people):
    match list_of_people:
        case []:
            print("Человек не найден.\n")
        case {'фамилия': f, 'имя': i, 'номер телефона': nt, 'дата рождения': dr}:
            print("\nИнформация о человеке:")
```

```

        print(f"Фамилия: {f}")
        print(f"Имя: {i}")
        print(f"Номер телефона: {nt}")
        print(f"Дата рождения: {dr}\n")

def save_to_json(file_name, list_of_people):
    """
    Сохранить всех в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        print(list_of_people)
        json.dump(list_of_people, fout, ensure_ascii=False, indent=4)
    print("Данные успешно сохранены в файл", file_name)

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

def main():
    list_of_people = []

    while True:
        print("\n1. Добавить человека")
        print("2. Найти человека по номеру телефона")
        print("3. Вывести список людей")
        print("4. Сохранить в json")
        print("5. Загрузить из json")
        print("6. Выйти")
        choice = input("Выберите действие (1/2/3/4/5/6): ")

        match choice:
            case '1':
                add_person(list_of_people)

            case '2':
                phone_to_find = input("Введите номер телефона для поиска: ")
                found_person = find_person_by_phone(list_of_people,
phone_to_find)
                print_person_info(found_person)

            case '3':
                for _ in list_of_people: print_person_info(_)

            case '4':
                file_name = str(input("Введите имя файла(без расширения): ")) +
'.json'
                save_to_json(file_name, list_of_people)

```

```

        case '5':
            file_name = str(input("Введите имя файла(без расширения): ")) +
'.json'

            list_of_people = load_from_json(file_name)
        case '6':
            print("Программа завершена.\n")
            break

        case _:
            print("Некорректный ввод. Попробуйте снова.\n")

if __name__ == "__main__":
    main()

```

Задание повышенной сложности

```

code > {} schema.json > ...
1  {
2      "$schema": "http://json-schema.org/draft-07/schema#",
3      "title": "Person",
4      "properties": {
5          "имя": {
6              "type": "string"
7          },
8          "фамилия": {
9              "type": "string"
10         },
11         "номер телефона": {
12             "type": "string"
13         },
14         "дата рождения": {
15             "type": "string"
16         }
17     },
18     "required": ["имя", "фамилия", "номер телефона", "дата рождения"],
19     "additionalProperties": false
20 }
21

```

Рисунок 5 – файл schema.json

```

68
69 def load_from_json(file_name):
70     """
71     Загрузить всех из JSON
72     """
73     with open(file_name, "r", encoding="utf-8") as fin:
74         document = json.load(fin)
75
76     if all(list(map(lambda x: check_validation_json(x), document))):
77         return document
78     else:
79         False
80
81 def check_validation_json(file_name):
82     with open('code/tasks/schema.json') as fs:
83         schema = json.load(fs)
84
85     try:
86         validate(instance=file_name, schema=schema)
87         return True
88     except ValidationError:
89         return False
90

```

Рисунок 6 – Проверка

Контрольные вопросы:

1. Для чего используется JSON? JSON – стандартный текстовый формат для хранения структурированных данных и обмена ими.
2. Какие типы значений используются в JSON? – В качестве значений в JSON могут быть использованы: запись (это неупорядоченное множество пар ключ:значение), массив, число, строка, true, false и null.
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? – JSON5 — это расширение популярного формата файлов JSON, которое упрощает написание и

поддержку вручную. Основные отличия JSON5 от оригинального формата JSON включают:

- 1) Поддержка комментариев;
- 2) Дополнительные типы данных;
- 3) В JSON5 запятые могут быть опущены после последнего элемента в массиве или объекте;
- 4) Необязательные кавычки для ключей;
- 5) Многострочные строки.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5? – работы с данными в формате JSON5 существуют сторонние библиотеки, которые предоставляют поддержку JSON5. Например `json5` или `rujson5`

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

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

`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? – 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

9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кириллицу? – Для поддержки кириллицы нужно установить параметр `ensure_ascii=False`

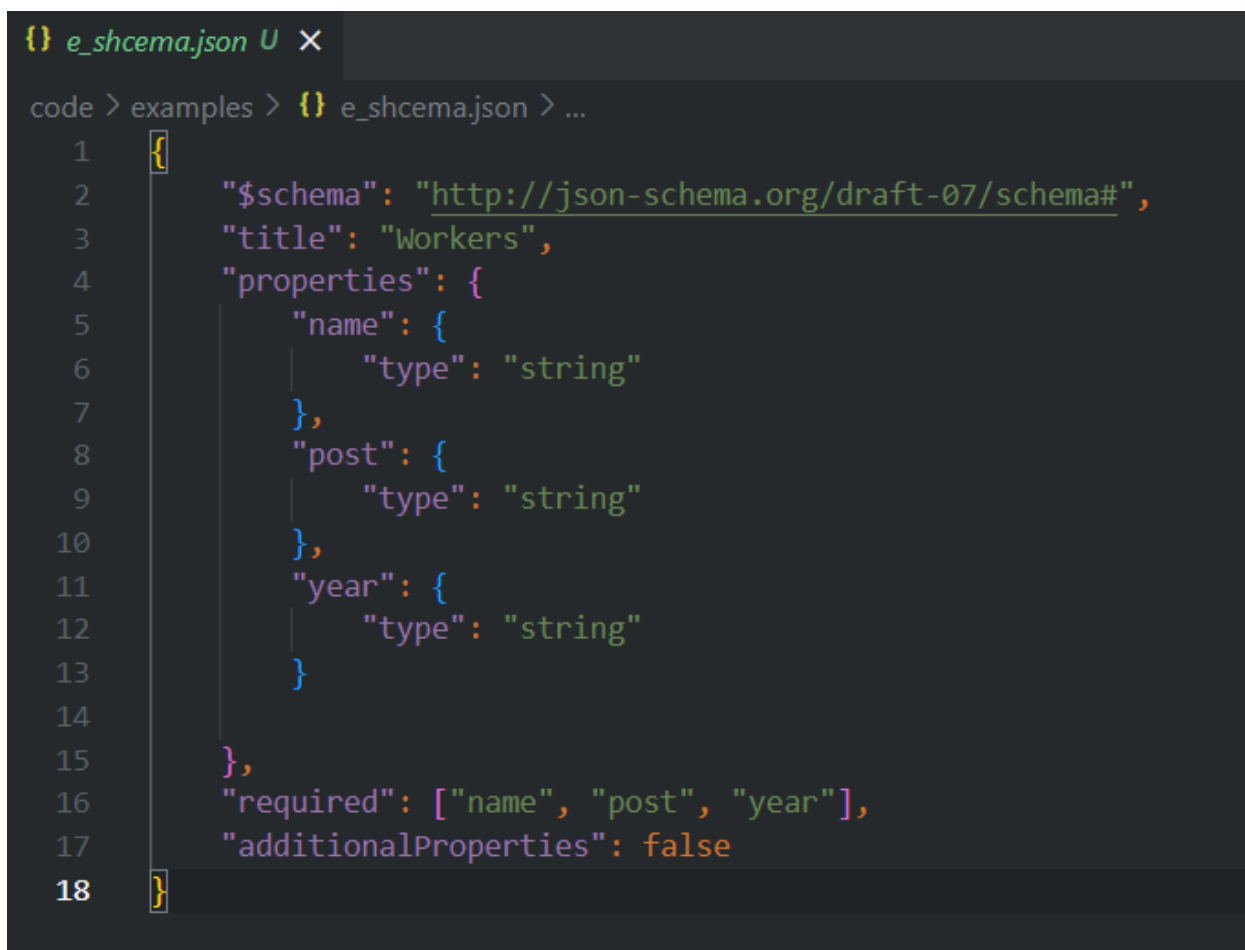
Пример:

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

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

это словарь, который можно использовать для аннотирования и проверки документов JSON.

Схема для примера 1:



```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Workers",
  "properties": {
    "name": {
      "type": "string"
    },
    "post": {
      "type": "string"
    },
    "year": {
      "type": "string"
    }
  },
  "required": ["name", "post", "year"],
  "additionalProperties": false
}
```

Рисунок 6 - файл e_schema.json