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

Халимендик Я. Д. « » 2023г.

Подпись студента_____

Работа защищена « »_____20__г.

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

Ход работы:

1. Изучить теоретический материал работы.

2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

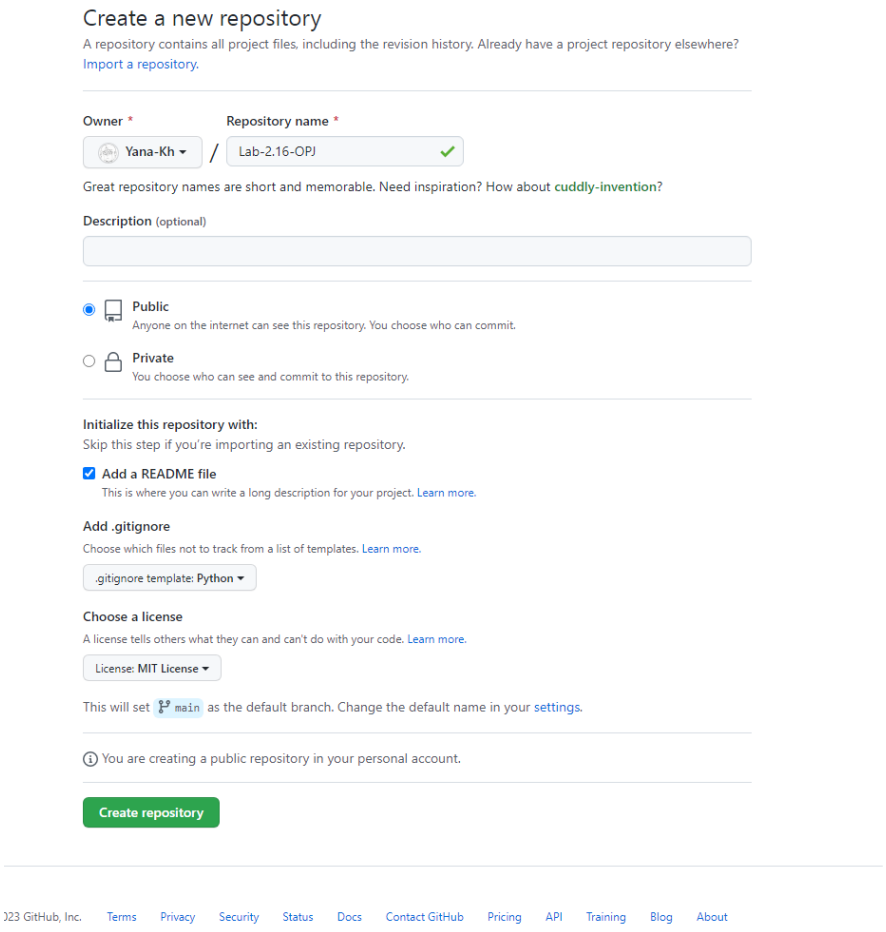

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-2.16-OPJ.git
Cloning into 'Lab-2.16-OPJ'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

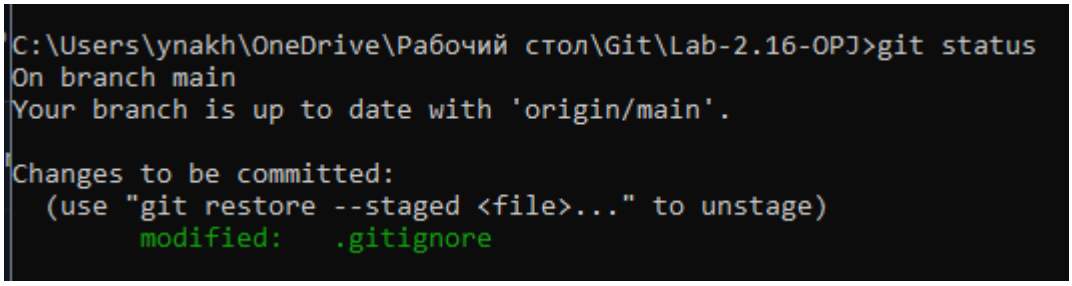

Рисунок 3 – Дополнение файла .gitignore

5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

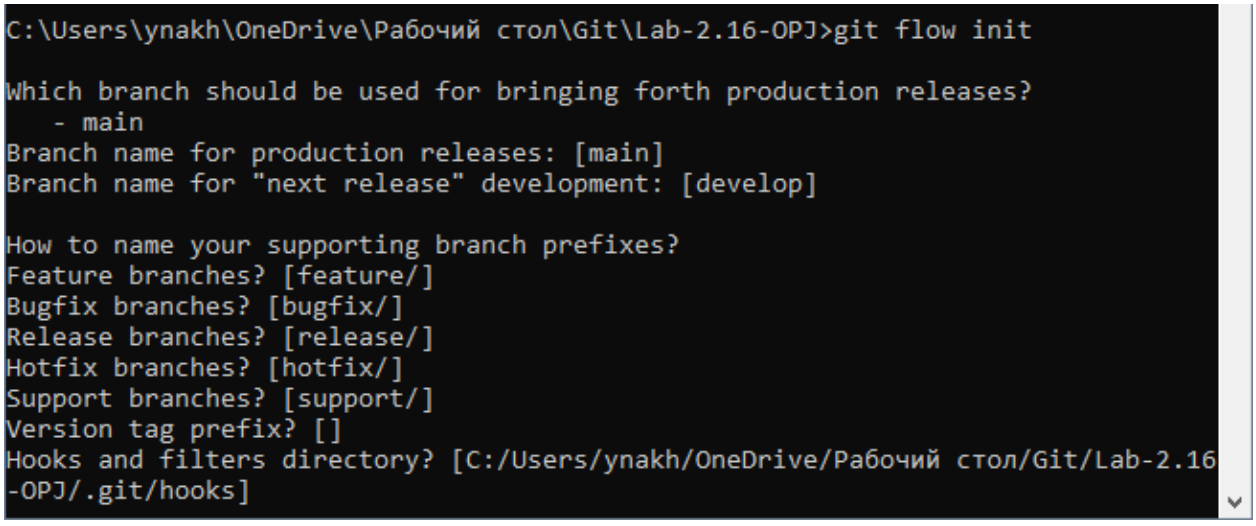

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

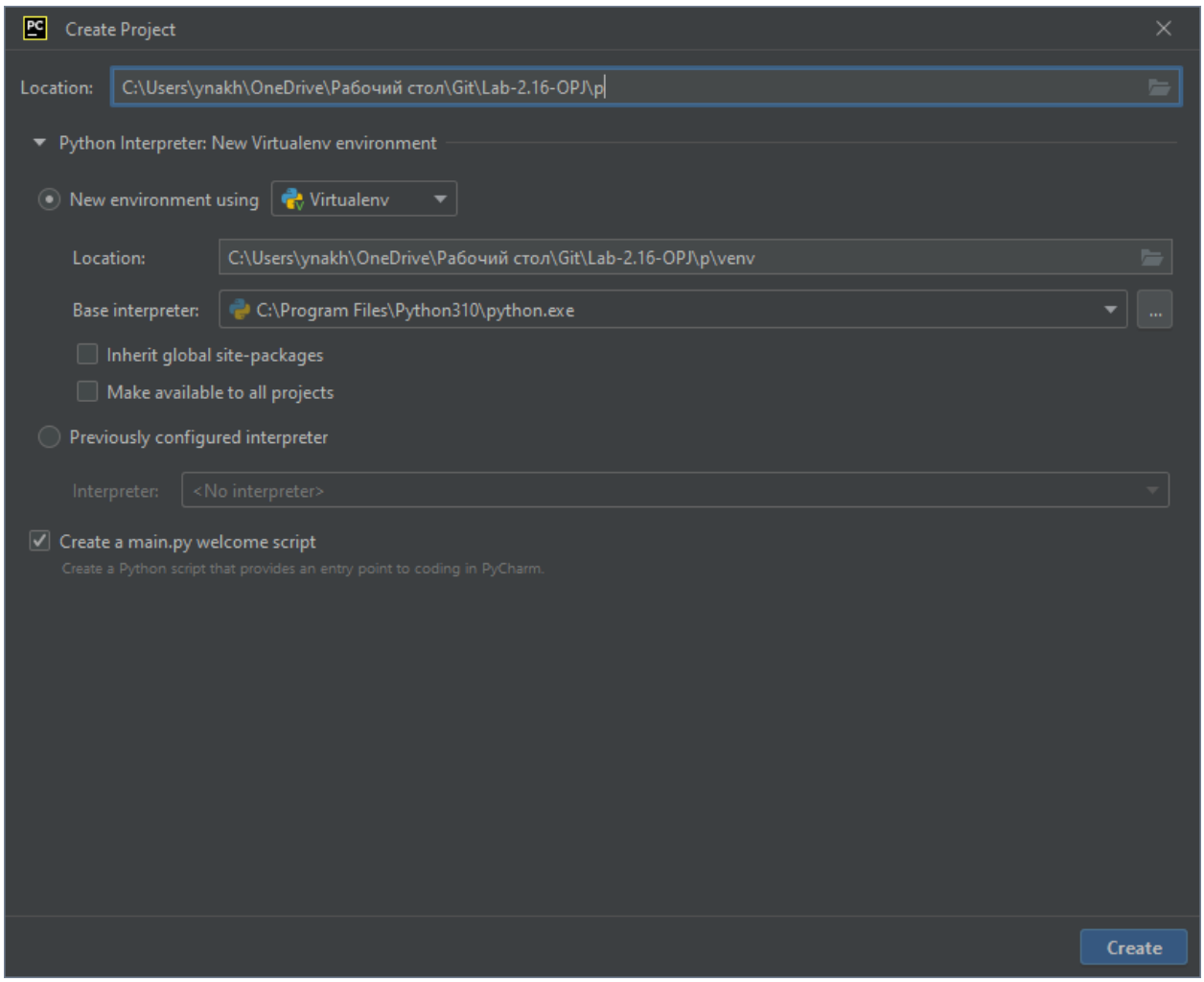

Рисунок 5 – Создание проекта PyCharm

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
```

```python
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)

def main():
    """
    Главная функция программы.
    """

    # Список работников.
    workers = []
    # 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 работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == "list":
            # Отобразить всех работников.
            display_workers(workers)

        elif command.startswith("select "):
            # Разбить команду на части для выделения стажа.
            parts = command.split(maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])

            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(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_workers(file_name, workers)

        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.
            workers = load_workers(file_name)

        elif command == 'help':
            # Вывести справку о работе с программой.
            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 - завершить работу с программой.")

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


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

```
>>> add
Фамилия и инициалы? Ключников ВЮ
Должность? учитель
Год поступления? 2020
>>> add
Фамилия и инициалы? Халимендик ЯД
Должность? студент
Год поступления? 2021
>>> add
Фамилия и инициалы? Попов ДС
Должность? учитель
Год поступления? 1999
>>> add
Фамилия и инициалы? Тулин ЕА
Должность? учитель
Год поступления? 2003
>>> add
Фамилия и инициалы? Душин ЛР
Должность? студент
Год поступления? 2009
>>> list
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Душин ЛР                       | студент              |     2009 |
|    2 | Ключников ВЮ                   | учитель              |     2020 |
|    3 | Попов ДС                       | учитель              |     1999 |
|    4 | Тулин ЕА                       | учитель              |     2003 |
|    5 | Халимендик ЯД                  | студент              |     2021 |
+------+--------------------------------+----------------------+----------+
>>> select 10
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Душин ЛР                       | студент              |     2009 |
|    2 | Попов ДС                       | учитель              |     1999 |
|    3 | Тулин ЕА                       | учитель              |     2003 |
+------+--------------------------------+----------------------+----------+
>>> save
>>> Неизвестная команда save
help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> save data
>>>
```

Рисунок 6 – Результат работы программы

8. Зафиксируйте сделанные изменения в репозитории.

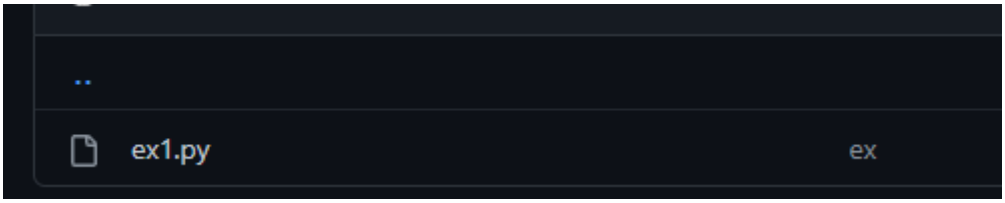

Рисунок 7 – Фиксирование изменений в репозитории

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.

Код:

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

import json
import sys
import datetime


def get_human():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и имя: ")
    phone = int(input("Номер телефона: +7"))
    bday = list(map(int, input("Дата рождения: ").split('.')))
    d_bday = datetime.date(bday[2], bday[1], bday[0])

    # Вернуть словарь.
    return {
        'name': name,
        'phone': phone,
        'birthday': d_bday
    }


def display_human(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
```

```
            '-' * 15,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^15} | {:^15} |'.format(
                "№",
                "Фамилия и имя",
                "Телефон",
                "День рождения"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, human in enumerate(staff, 1):
            print(
                f'| {idx:>4} |'
                f' {human.get("name", ""):<30} |'
                f' {human.get("phone", 0):<15} |'
                f' {human.get("birthday")}      |'
            )
            print(line)

    else:
        print("Список пуст.")


def find_human(staff, fname):
    """
    Выбрать работников с заданным стажем.
    """
    # Сформировать список людей.
    result = []
    for h in staff:
        if fname in str(h.values()):
            result.append(h)

    # Проверка на наличие записей
    if len(result) == 0:
        return print("Запись не найдена")

    # Возвратить список выбранных работников.
    return result


def json_deserial(obj):
    """
    Деериализация объектов datetime
    """
    for h in obj:
        if isinstance(h['birthday'], str):
            #print(datetime.strptime(h['birthday'], '%Y-%m-%d'))
            bday = list(map(int, h['birthday'].split('-')))
            h['birthday'] = datetime.date(bday[0], bday[1], bday[2])


def load_humans(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)


def json_serial(obj):
    """Сериализация объектов datetime"""

    if isinstance(obj, (datetime.datetime, datetime.date)):
        return obj.isoformat()


def save_humans(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,
default=json_serial)


def main():
    """
    Главная функция программы.
    """
    # Список работников.
    people = []

    # 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 работнике.
            human = get_human()

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

        elif command == 'list':
            # Отобразить всех работников.
            display_human(people)

        elif command == 'find':
            f = input('Введите фамилию: ')

            # Выбрать людей с заданной фамилией.
            selected = find_human(people, f)
            # Отобразить выбранных работников.
            display_human(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_humans(file_name, people)

        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.
            people = load_humans(file_name)
            json_deserial(people)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить человека;")
            print("list - вывести список людей;")
            print("find - найти человека по фамилии;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

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


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

```
>>> add
Фамилия и имя: Халимендик Яна
Номер телефона: +79188800809
Дата рождения: 05.05.2003
>>> add
Фамилия и имя: Ключников Вячеслав
Номер телефона: +79563742134
Дата рождения: 03.11.1995
>>> add
Фамилия и имя: Иванов Петр
Номер телефона: +79179097727
Дата рождения: 12.10.2009
>>> list
+------+--------------------------------+-----------------+-----------------+
|  №   |         Фамилия и имя          |     Телефон     |  День рождения  |
+------+--------------------------------+-----------------+-----------------+
|    1 | Иванов Петр                    | 9179097727      | 2009-10-12      |
+------+--------------------------------+-----------------+-----------------+
|    2 | Халимендик Яна                 | 9188800809      | 2003-05-05      |
+------+--------------------------------+-----------------+-----------------+
|    3 | Ключников Вячеслав             | 9563742134      | 1995-11-03      |
+------+--------------------------------+-----------------+-----------------+
>>> save data_id
>>> load data_id
>>> list
+------+--------------------------------+-----------------+-----------------+
|  №   |         Фамилия и имя          |     Телефон     |  День рождения  |
+------+--------------------------------+-----------------+-----------------+
|    1 | Иванов Петр                    | 9179097727      | 2009-10-12      |
+------+--------------------------------+-----------------+-----------------+
|    2 | Халимендик Яна                 | 9188800809      | 2003-05-05      |
+------+--------------------------------+-----------------+-----------------+
|    3 | Ключников Вячеслав             | 9563742134      | 1995-11-03      |
+------+--------------------------------+-----------------+-----------------+
>>> find Петров
>>> Неизвестная команда find петров
find
Введите фамилию: Петров
Запись не найдена
Список пуст.
>>> find
Введите фамилию: Ключников
+------+--------------------------------+-----------------+-----------------+
|  №   |         Фамилия и имя          |     Телефон     |  День рождения  |
+------+--------------------------------+-----------------+-----------------+
|    1 | Ключников Вячеслав             | 9563742134      | 1995-11-03      |
+------+--------------------------------+-----------------+-----------------+
>>>
```

Рисунок 8 – Результат работы 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.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import json
import sys
from datetime import date
from jsonschema import validate

schema = {

        "name": {"type": "string"},
        "post": {"type": "string"},
        "year": {"type": "number"},

}


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


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)


def main():
```

```
    """
    Главная функция программы.
    """

    # Список работников.
    workers = []
    # 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 работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get("name", ""))

        elif command == "list":
            # Отобразить всех работников.
            display_workers(workers)

        elif command.startswith("select "):
            # Разбить команду на части для выделения стажа.
            parts = command.split(maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])

            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(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_workers(file_name, workers)

        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.
            workers = load_workers(file_name)
            validate(workers, schema=schema)

        elif command == "help":
            # Вывести справку о работе с программой.
            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 - завершить работу с программой.")

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


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

10. Зафиксируйте сделанные изменения в репозитории.

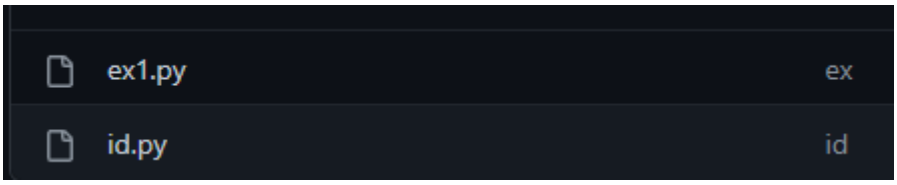

Рисунок 9 – Фиксирование изменений в репозитории

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.