

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 №2.16
дисциплины «Основы программной инженерии»

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

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

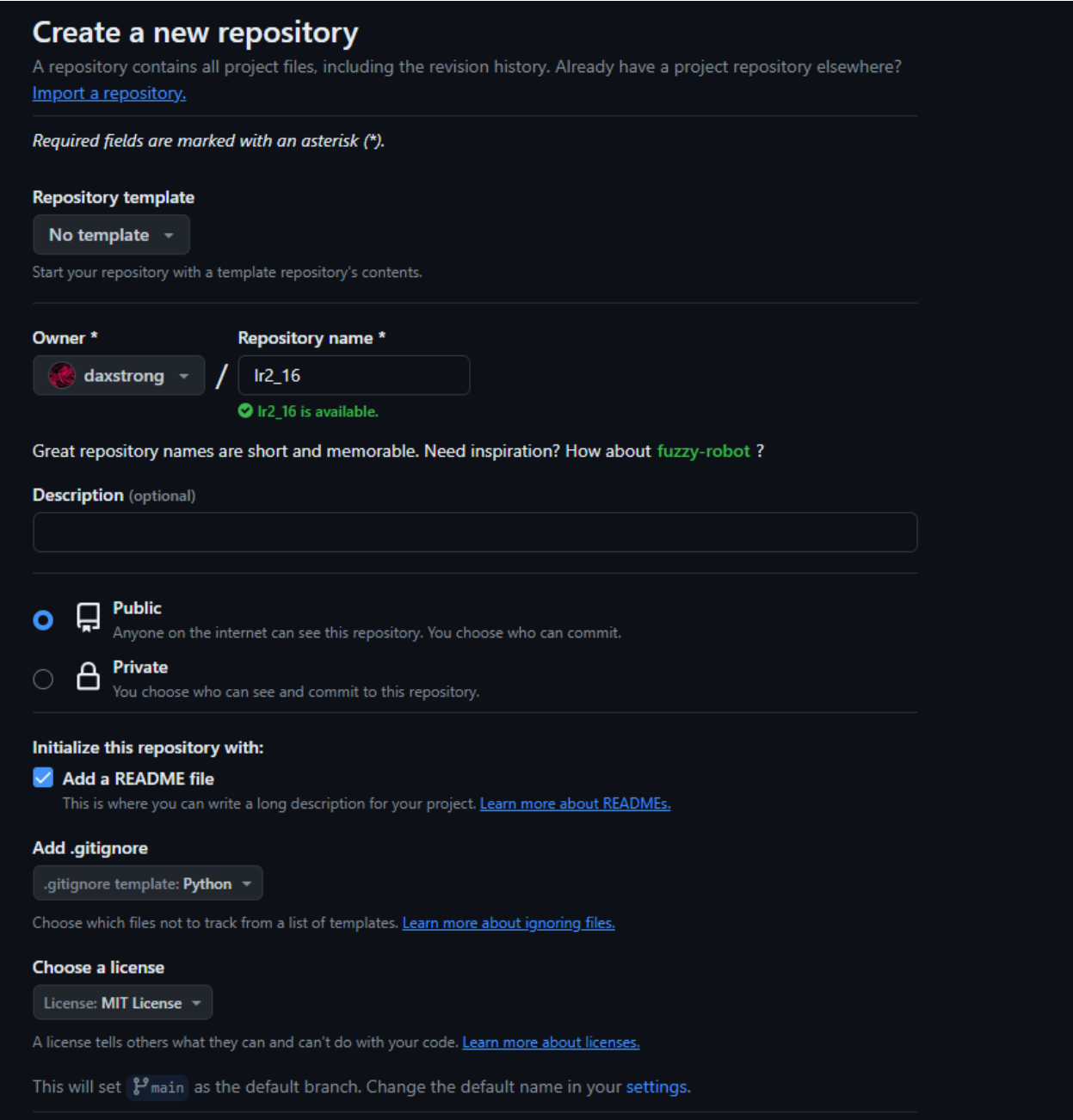
Ставрополь, 2023 г.

Тема: Работа с данными формата JSON в языке Python.

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

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * daxstrong ▾ / **Repository name *** lr2_16

✔ lr2_16 is available.

Great repository names are short and memorable. Need inspiration? How about **fuzzy-robot** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Рисунок 1 – Создание репозитория с заданными настройками

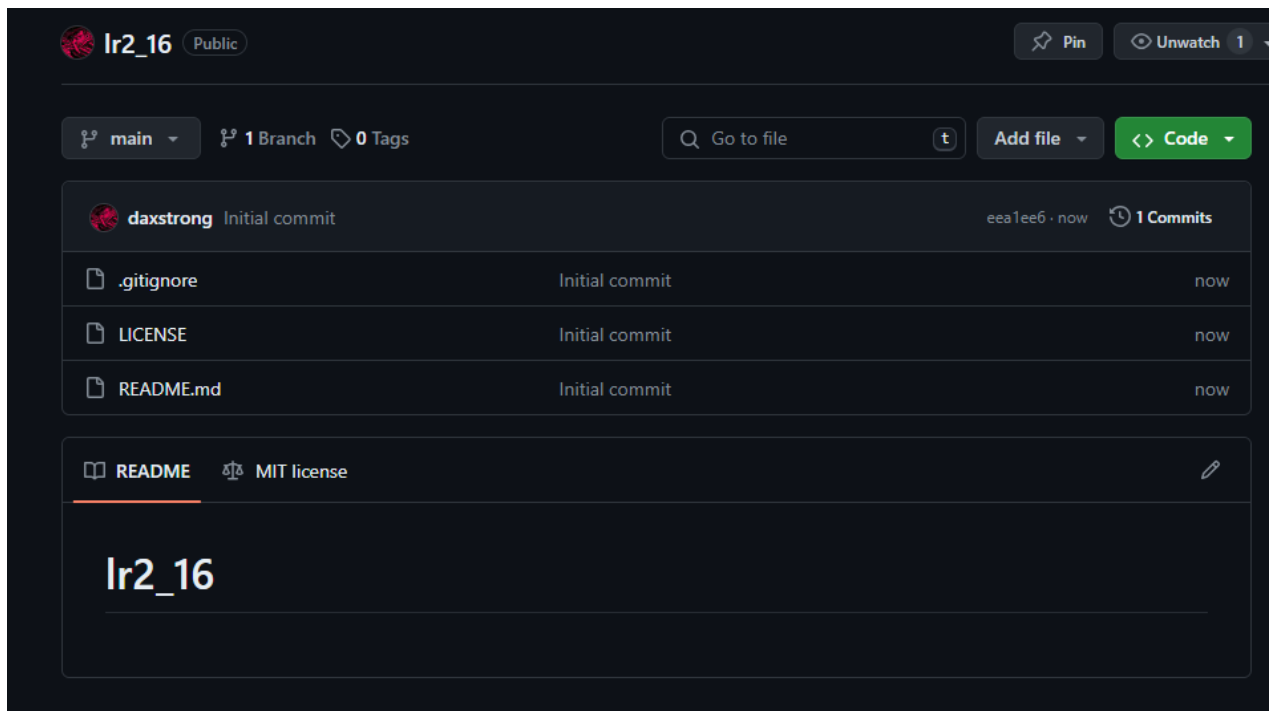


Рисунок 2 – Созданный репозиторий

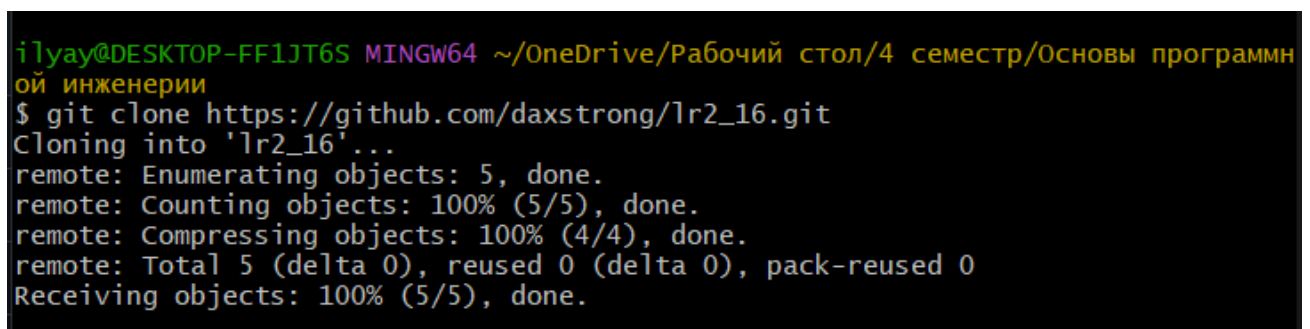


Рисунок 3 – Клонирование репозитория

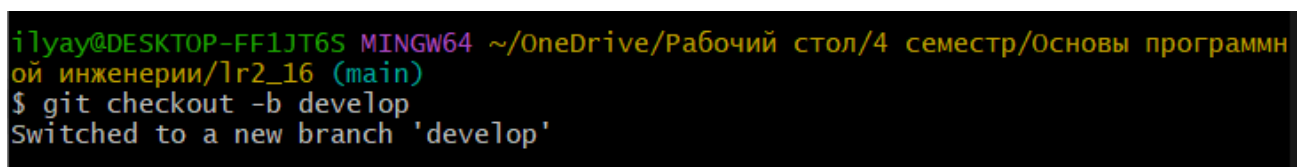


Рисунок 4 – Создание ветки develop

```
(base) PS C:\Users\ilyay> cd "C:\Users\ilyay\OneDrive\Рабочий стол\4 семестр\Основы программной инженерии\lr2_16"
(base) PS C:\Users\ilyay\OneDrive\Рабочий стол\4 семестр\Основы программной инженерии\lr2_16> conda create -n lr2_14 python=3.11.7
Retrieving notices: ...working... done
WARNING: A conda environment already exists at 'C:\Users\ilyay\anaconda3\envs\lr2_14'
Remove existing environment (y/[n])? y
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.3.0

Please update conda by running

  $ conda update -n base -c defaults conda

or to minimize the number of packages updated during conda update use

  conda install conda=24.3.0

## Package Plan ##

  environment location: C:\Users\ilyay\anaconda3\envs\lr2_14
  added / updated specs:
    - python=3.11.7

The following packages will be downloaded:



| package                   | build      | size          |
|---------------------------|------------|---------------|
| bzip2-1.0.8               | h2bbff1b_5 | 78 KB         |
| ca-certificates-2024.3.11 | haa95532_0 | 128 KB        |
| tzdata-2024a              | h04d1e81_0 | 116 KB        |
| xz-5.4.6                  | h8cc25b3_0 | 587 KB        |
| <b>Total:</b>             |            | <b>910 KB</b> |



The following NEW packages will be INSTALLED:

bzip2                pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_5
ca-certificates      pkgs/main/win-64::ca-certificates-2024.3.11-haa95532_0
libffi               pkgs/main/win-64::libffi-3.4.4-hd77b12b_0
openssl              pkgs/main/win-64::openssl-3.0.13-h2bbff1b_0
pip                  pkgs/main/win-64::pip-23.3.1-py311haa95532_0
```

Рисунок 5 – Создание виртуального окружения с помощью Anaconda

2. Проработать пример лабораторной работы. Модифицировать пример №1 из лабораторной работы №2.8 и добавить возможность сохранения списка:

```
#!/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

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 кириллицы 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 соответствии 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()

```

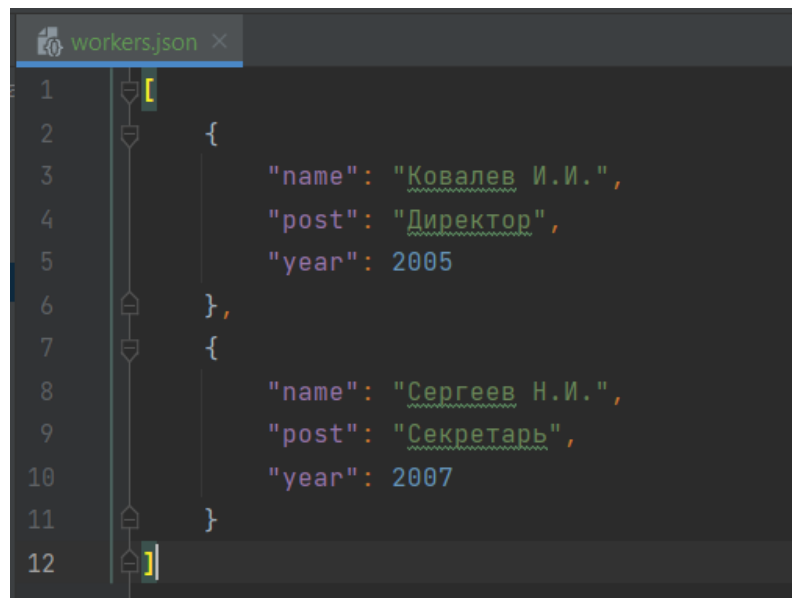
Листинг 1 – Пример с новыми командами и методами (сохранение и выгрузка информации о работниках)

```

C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий стол
>>> add
Фамилия и инициалы? Ковалев И.И.
Должность? Директор
Год поступления? 2005
>>> add
Фамилия и инициалы? Сергеев Н.И.
Должность? Секретарь
Год поступления? 2007
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Ковалев И.И. | Директор | 2005 |
| 2 | Сергеев Н.И. | Секретарь | 2007 |
+-----+-----+-----+-----+
>>> save workers.json
>>>

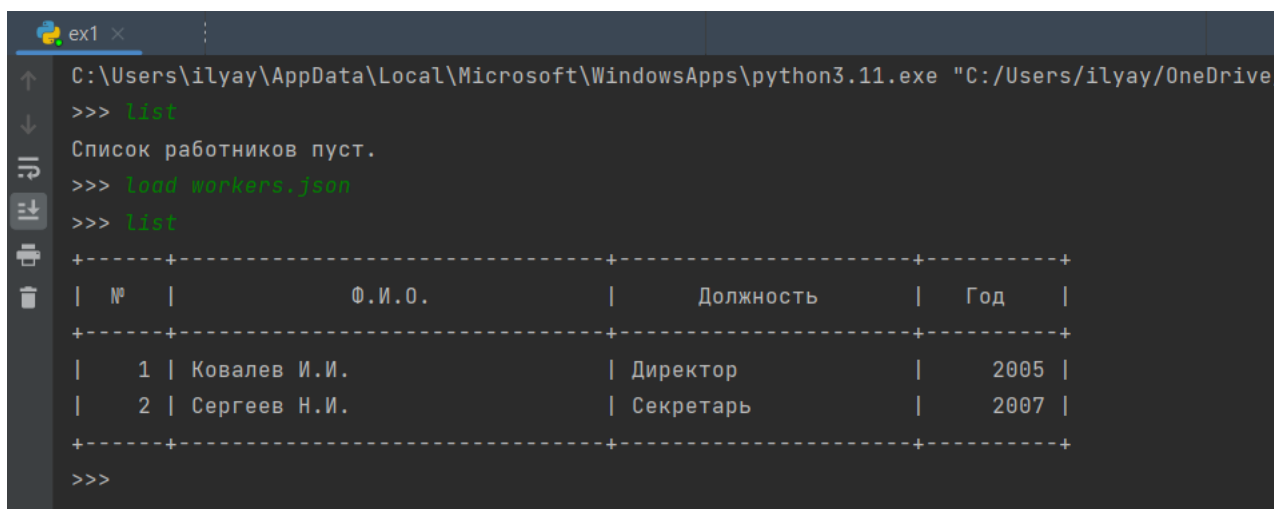
```

Рисунок 6 – Добавление и сохранение информации о работниках в файле workers.json



```
1  [
2      {
3          "name": "Ковалев И.И.",
4          "post": "Директор",
5          "year": 2005
6      },
7      {
8          "name": "Сергеев Н.И.",
9          "post": "Секретарь",
10         "year": 2007
11     }
12 ]
```

Рисунок 7 – Содержимое файла workers.json



```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive
>>> list
Список работников пуст.
>>> load workers.json
>>> list
```

№	Ф.И.О.	Должность	Год
1	Ковалев И.И.	Директор	2005
2	Сергеев Н.И.	Секретарь	2007

```
>>>
```

Рисунок 8 – Загрузка информации из json файла

3. Выполним индивидуальные задания:

Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON:

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

import json
import sys
```



```

from datetime import datetime

def exit_program():
    """
    Выход из программы.
    """
    sys.exit()

def add_person(people):
    """
    Добавление информации о человеке.
    """
    last_name = input("Фамилия: ")
    first_name = input("Имя: ")
    phone_number = input("Номер телефона: ")
    birthdate_str = input("Дата рождения (в формате ДД.ММ.ГГГГ): ")
    birthdate = datetime.strptime(birthdate_str, "%d.%m.%Y")

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

    people.append(person)
    people.sort(key=lambda x: x['фамилия'])

def list_people(people):
    """
    Вывод списка всех людей.
    """
    line = f'+-{"-" * 25}-+-{"-" * 15}-+-{"-" * 25}-+'
    print(line)
    print(f"| {'Фамилия':^25} | {'Имя':^15} | {'Дата рождения':^25}"
          |")

    for person in people:
        print(line)
        print(f"| {person['фамилия']:^25} | {person['имя']:^15} |"
              {person['дата рождения'].strftime('%d.%m.%Y'):^25} |")
        print(line)

def select_people_by_month(people, month_to_search):
    """
    Вывод людей с днем рождения в указанном месяце.
    """
    found = False

    print(f"Люди с днем рождения в месяце {month_to_search}:")
    for person in people:
        if person['дата рождения'].month == month_to_search:
            print(

```

```

        f"Фамилия: {person['фамилия']}, Имя: {person['имя']},
Дата рождения: {person['дата рождения'].strftime('%d.%m.%Y')}"
        found = True

    if not found:
        print("Нет людей с днем рождения в указанном месяце.")

def help_info():
    """
    Вывод справочной информации о командах.
    """
    print("Список команд:\n")
    print("add - добавить информацию о человеке;")
    print("list - вывести список всех людей;")
    print("select <месяц> - вывести людей с днем рождения в указанном
месяце;")
    print("save - сохранить данные в файл JSON;")
    print("load - загрузить данные из файла JSON;")
    print("exit - завершить работу с программой.")

def save_people(file_name, people):
    """
    Сохранить всех людей в файл JSON.
    """
    # Преобразуем объекты datetime в строки
    for person in people:
        person['дата рождения'] = person['дата
рождения'].strftime('%d.%m.%Y')

    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(people, fout, ensure_ascii=False, indent=4)

def load_people(file_name):
    """
    Загрузить всех людей из файла JSON.
    """
    try:
        with open(file_name, "r", encoding="utf-8") as fin:
            people_data = json.load(fin)
            for person in people_data:
                person['дата рождения'] =
datetime.strptime(person['дата рождения'], '%d.%m.%Y')
            return people_data
    except FileNotFoundError:
        return []

if __name__ == '__main__':
    file_name = 'people.json' # Имя файла для сохранения данных

    people = load_people(file_name)

    while True:

```

```

command = input(">>> ").lower()

match command:
    case 'exit':
        save_people(file_name, people)  # Сохраняем данные
        exit_program()
    case 'add':
        add_person(people)
    case 'list':
        list_people(people)
    case command if command.startswith('select '):
        month_to_search = int(command.split(' ')[1])
        select_people_by_month(people, month_to_search)
    case command if command.startswith('save '):
        save_file_name = command.split(' ')[1]
        save_people(save_file_name, people)
    case command if command.startswith('load '):
        load_file_name = command.split(' ')[1]
        people = load_people(load_file_name)
    case 'help':
        help_info()
    case _:
        print(f"Неизвестная команда {command}",
file=sys.stderr)

```

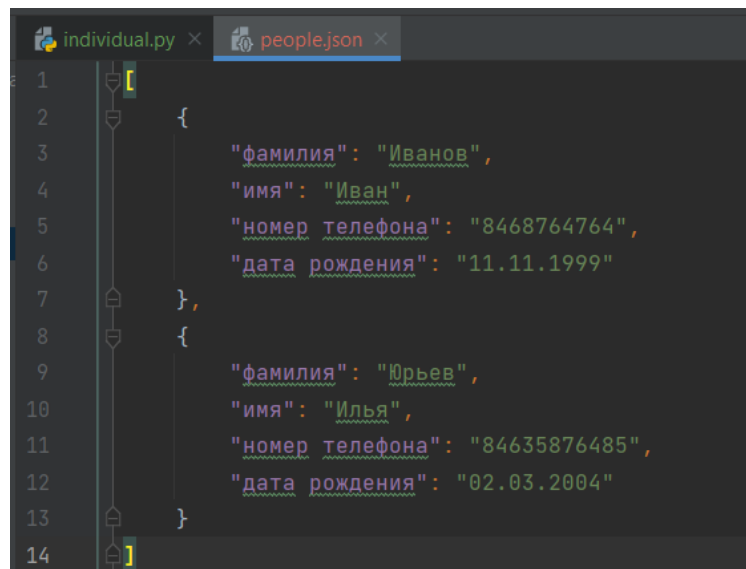
Листинг 2 – Переделанный пример индивидуального задания лабораторной работы №2.8. Добавлены функции `save_people` и `load_people`, чтобы обеспечить сохранение и загрузку данных в файл JSON

```

C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий стол/
>>> add
Фамилия: Юрьев
Имя: Илья
Номер телефона: 84635876485
Дата рождения (в формате ДД.ММ.ГГГГ): 02.03.2004
>>> add
Фамилия: Иванов
Имя: Иван
Номер телефона: 8468764764
Дата рождения (в формате ДД.ММ.ГГГГ): 11.11.1999
>>> list
+-----+-----+-----+
|      Фамилия      |      Имя      |      Дата рождения      |
+-----+-----+-----+
|      Иванов      |      Иван      |      11.11.1999      |
+-----+-----+-----+
|      Юрьев      |      Илья      |      02.03.2004      |
+-----+-----+-----+
>>> save people.json
>>>

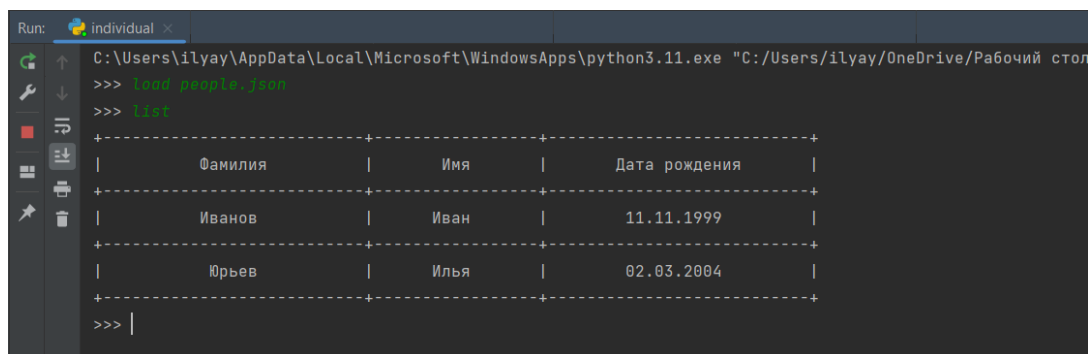
```

Рисунок 9 – Добавление информации о людях и сохранение в файл `people.json`



```
1 [
2   {
3     "фамилия": "Иванов",
4     "имя": "Иван",
5     "номер телефона": "8468764764",
6     "дата рождения": "11.11.1999"
7   },
8   {
9     "фамилия": "Юрьев",
10    "имя": "Илья",
11    "номер телефона": "84635876485",
12    "дата рождения": "02.03.2004"
13  }
14 ]
```

Рисунок 10 – Сохраненные данные в people.json



```
Run: individual.py
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий стол
>>> load_people.json
>>> list
+-----+-----+-----+
| Фамилия | Имя | Дата рождения |
+-----+-----+-----+
| Иванов | Иван | 11.11.1999 |
+-----+-----+-----+
| Юрьев | Илья | 02.03.2004 |
+-----+-----+-----+
>>> |
```

Рисунок 11 – Загрузка информации о людях из файла people.json

4. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:



```
$ git log
commit bb445339a01a2871eec1a83301f4cd6149fd79ce (HEAD -> develop)
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date: Wed Apr 17 01:38:51 2024 +0300

    final changes

commit eealee6532126fafba39904b8ce5a0eda9483f51 (origin/main, origin/HEAD, main)
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date: Tue Apr 16 22:42:50 2024 +0300

    Initial commit

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программн
ой инженерии/lr2_16 (develop)
$
```

Рисунок 12 – Коммиты ветки develop во время выполнения лабораторной Работы

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программн
ой инженерии/lr2_16 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программной инженерии/lr2_16 (main)
$ git merge develop
Updating eealee6..bb44533
Fast-forward
 .idea/.gitignore | 8 ++
 .idea/inspectionProfiles/Project_Default.xml | 51 +++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +
 .idea/lr2_16.iml | 8 ++
 .idea/misc.xml | 4 +
 .idea/modules.xml | 8 ++
 .idea/vcs.xml | 6 +
 ex1.py | 162 +++++++++++++++++++++++++++++++++++++
 individual.py | 135 +++++++++++++++++++++++++++++++++++++
9 files changed, 388 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/Project_Default.xml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lr2_16.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 ex1.py
create mode 100644 individual.py

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программной инженерии/lr2_16 (main)
$

```

Рисунок 13 – Слияние веток main и develop

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программной инженерии/lr2_16 (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 5.54 KiB | 2.77 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/daxstrong/lr2_16.git
 eealee6..bb44533 main -> main

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/4 семестр/Основы программной инженерии/lr2_16 (main)
$ |

```

Рисунок 14 – Отправка изменений на удаленный репозиторий

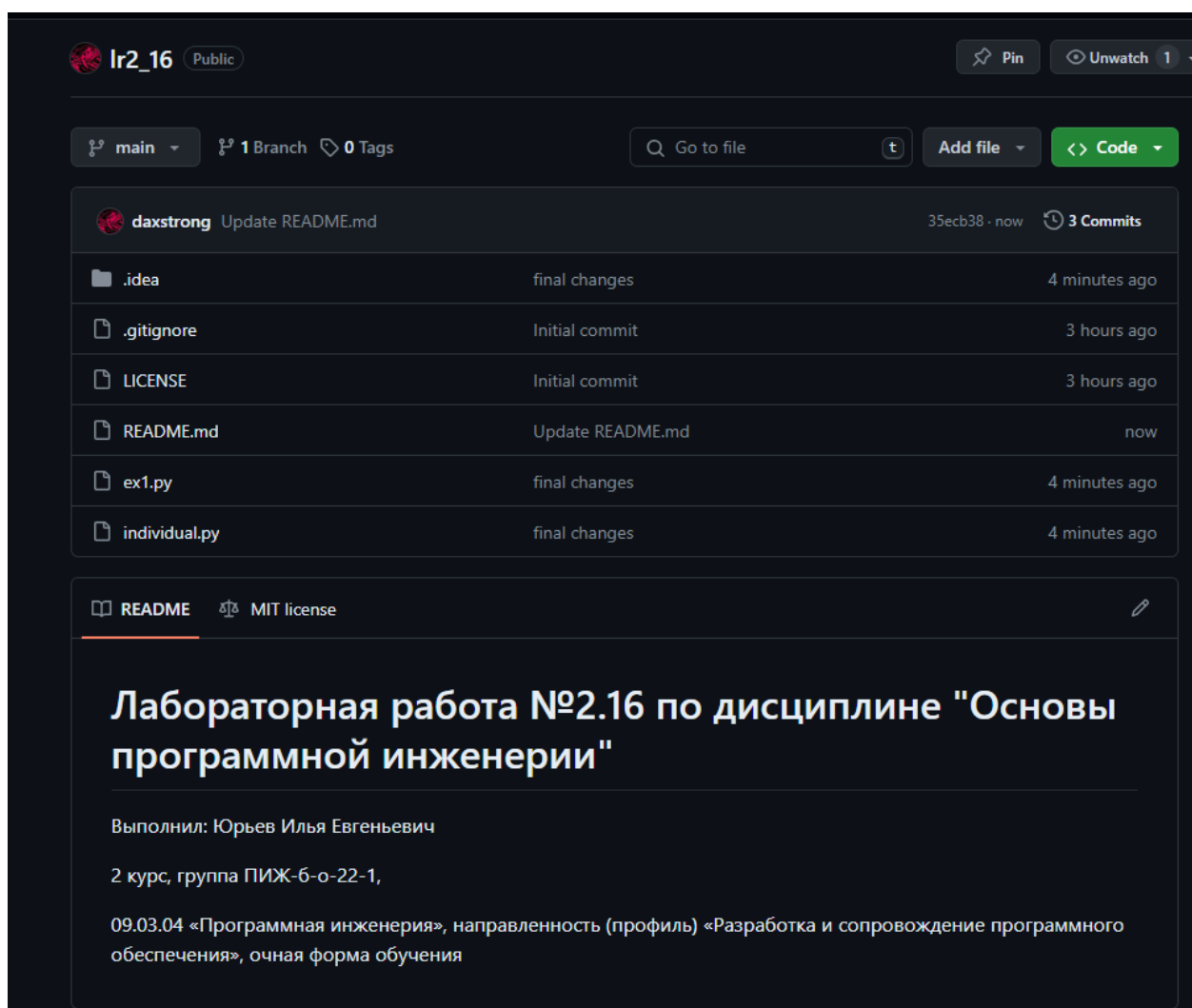


Рисунок 15 – Изменения удаленного репозитория

Ссылка на репозиторий **GitHub**: https://github.com/daxstrong/lr2_16

Ответы на контрольные вопросы:

1. Для чего используется JSON?

JSON (JavaScript Object Notation) используется для обмена данными между приложениями. Он предоставляет простой и удобный формат для хранения и передачи структурированных данных.

2. Какие типы значений используются в JSON?

В JSON могут использоваться следующие типы значений: строки, числа, булевы значения, массивы, объекты, null.

3. Как организована работа со сложными данными в JSON?

В JSON сложные данные организованы в виде вложенных объектов и массивов. Это позволяет представлять структурированные данные различной сложности, включая вложенные структуры и списки объектов.

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

JSON5 – это расширение JSON, которое добавляет некоторые дополнительные возможности, такие как поддержка комментариев, необязательные запятые в конце списков и ключей объектов, а также поддержка многострочных строк. Основное отличие от JSON заключается в том, что JSON5 более гибок и удобен для чтения и написания людьми.

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 сторонние библиотеки, такие как `json5` или `json5-data`, которые позволяют работать с данными в формате JSON5 так же, как и с данными в формате JSON.

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

В Python для сериализации данных в формат JSON используется модуль `json`. Он предоставляет функцию `json.dump()` для записи данных в файл и функцию `json.dumps()` для преобразования данных в строку JSON.

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

`json.dump()` записывает данные в файл, в то время как `json.dumps()` возвращает строку JSON. Таким образом, `json.dump()` принимает два аргумента: данные и файловый объект для записи, в то время как `json.dumps()` принимает только один аргумент - данные, которые нужно преобразовать в JSON строку.

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

В Python для десериализации данных из формата JSON используется модуль `json`. Он предоставляет функцию `json.load()` для чтения данных из файла и функцию `json.loads()` для преобразования строки JSON в объект Python.

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

Для работы с данными формата JSON, содержащими кириллицу, в Python можно использовать параметр `ensure_ascii=False` при сериализации данных с помощью `json.dump()` или `json.dumps()`. Это позволяет сохранить кириллические символы в их оригинальном виде без преобразования в ASCII.