

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с данными формата JSON в языке Python»

ОТЧЕТ
по лабораторной работе №19
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

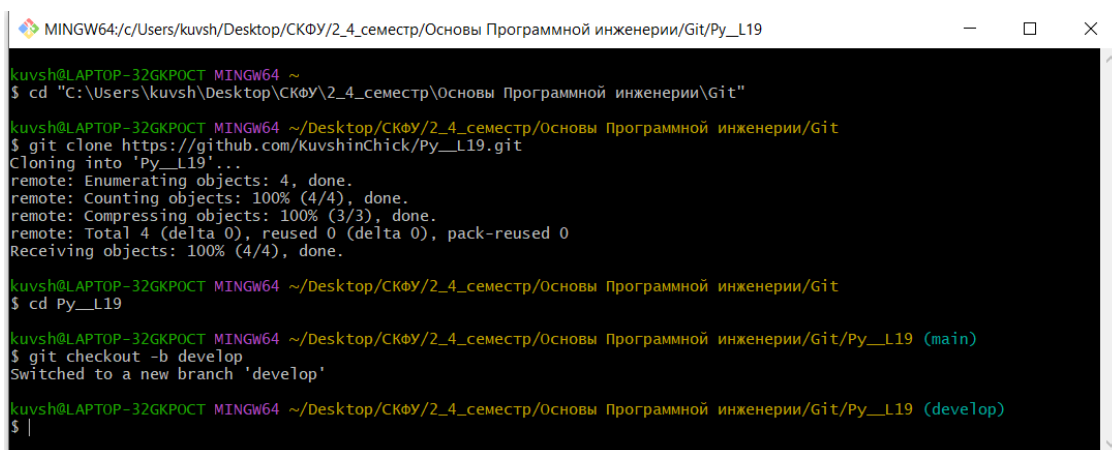
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

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

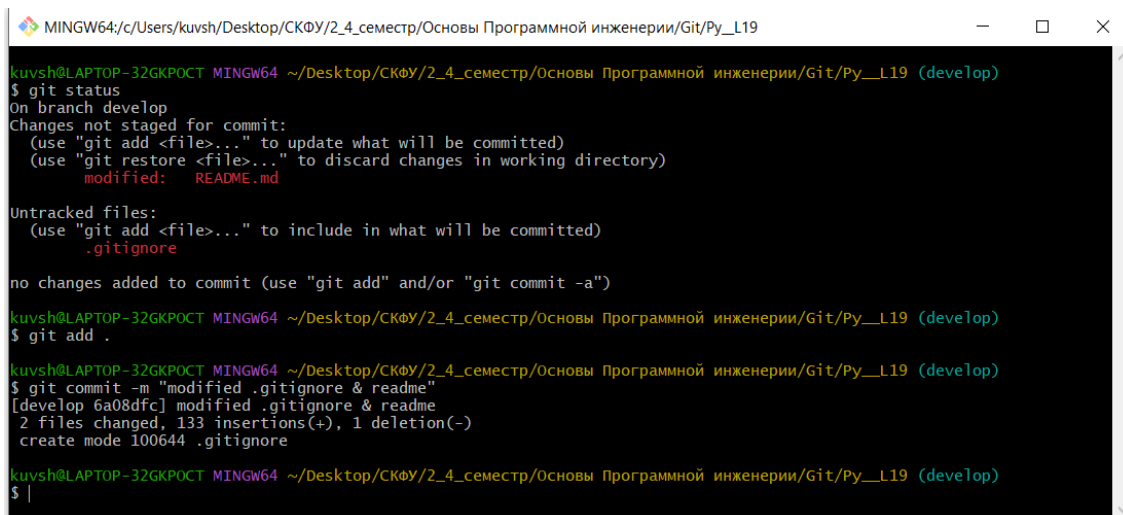
Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT 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 соответствии с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Основы Программной инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ git clone https://github.com/KuvshinChick/Py_L19.git
Cloning into 'Py_L19'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ cd Py_L19
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (develop)
$ |
```

Рисунок 19.1 – Клонирование репозитория и создание ветки develop



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (develop)
$ git add .
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 6a08dfc] modified .gitignore & readme
 2 files changed, 133 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py_L19 (develop)
$ |
```

Рисунок 19.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.

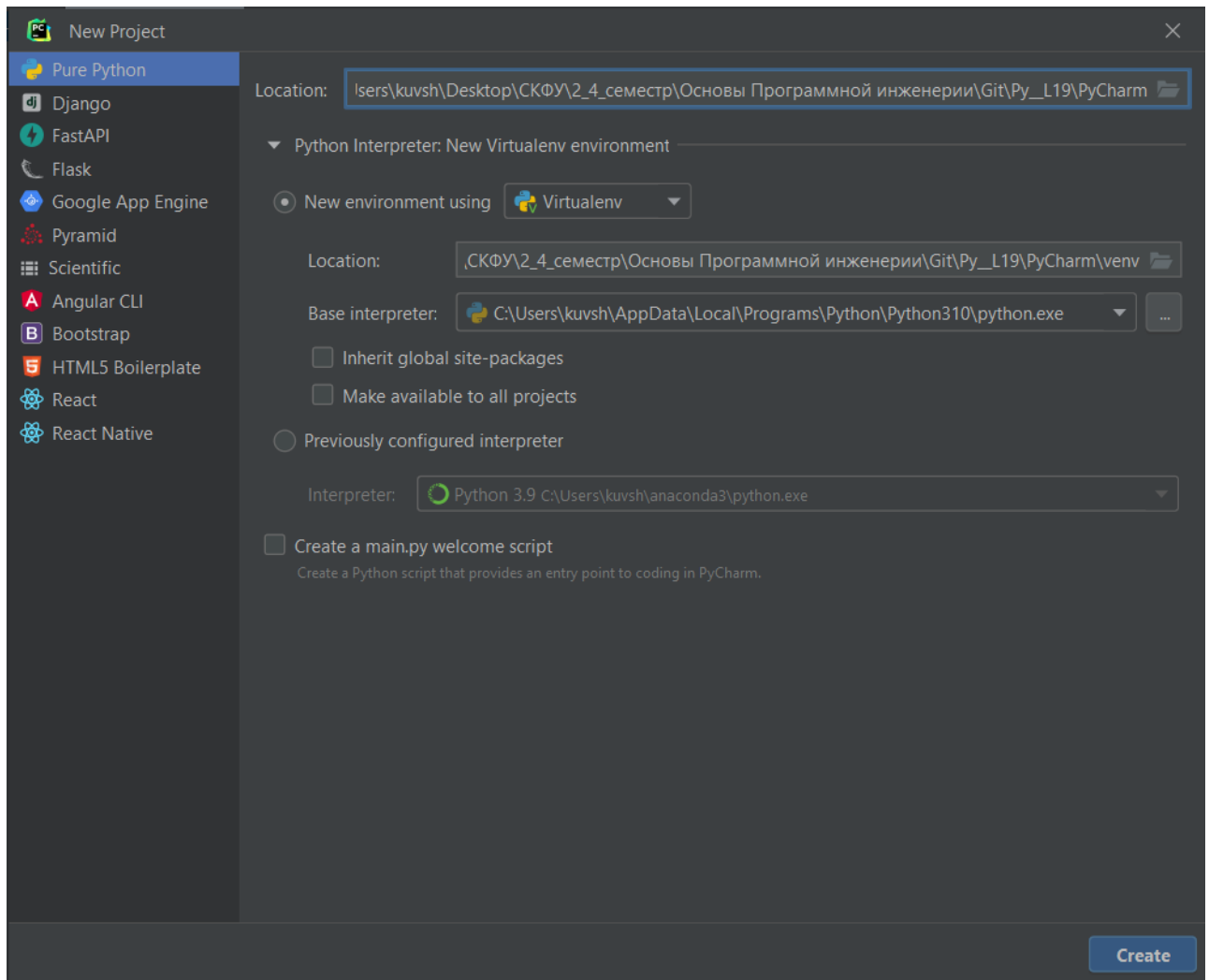


Рисунок 19.3 – Создание проекта и виртуального окружения

7. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории

8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

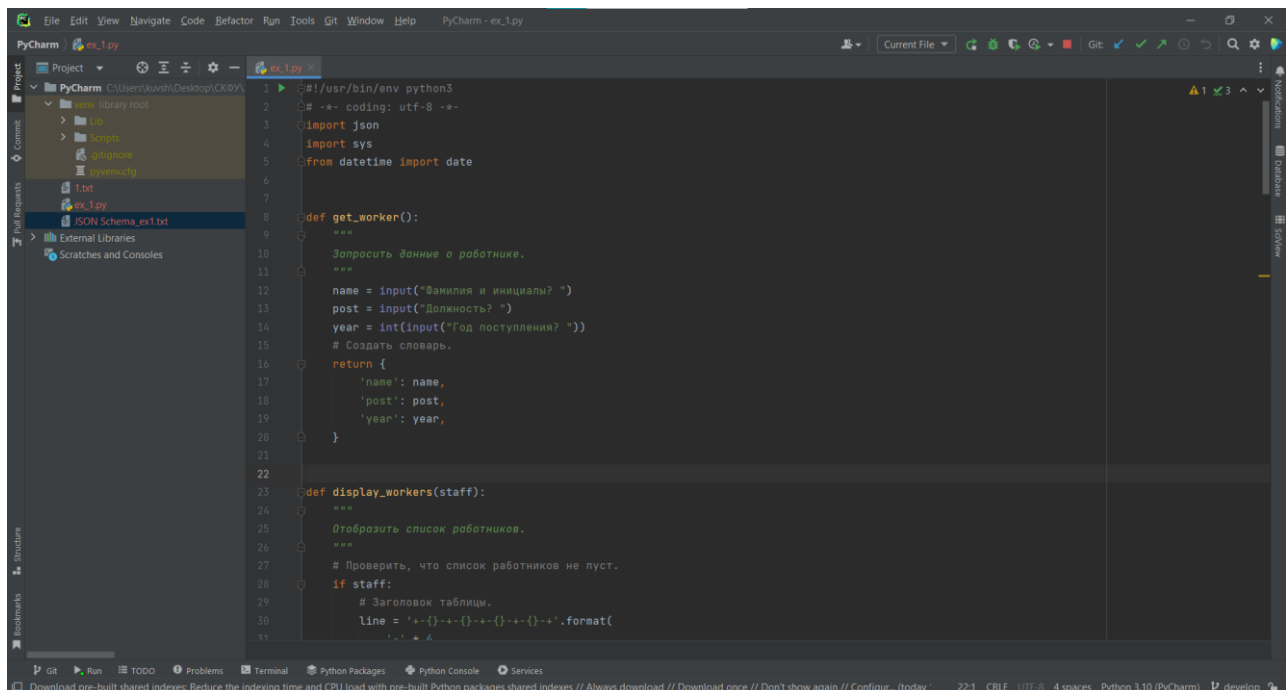


Рисунок 19.4 – Проработка примера

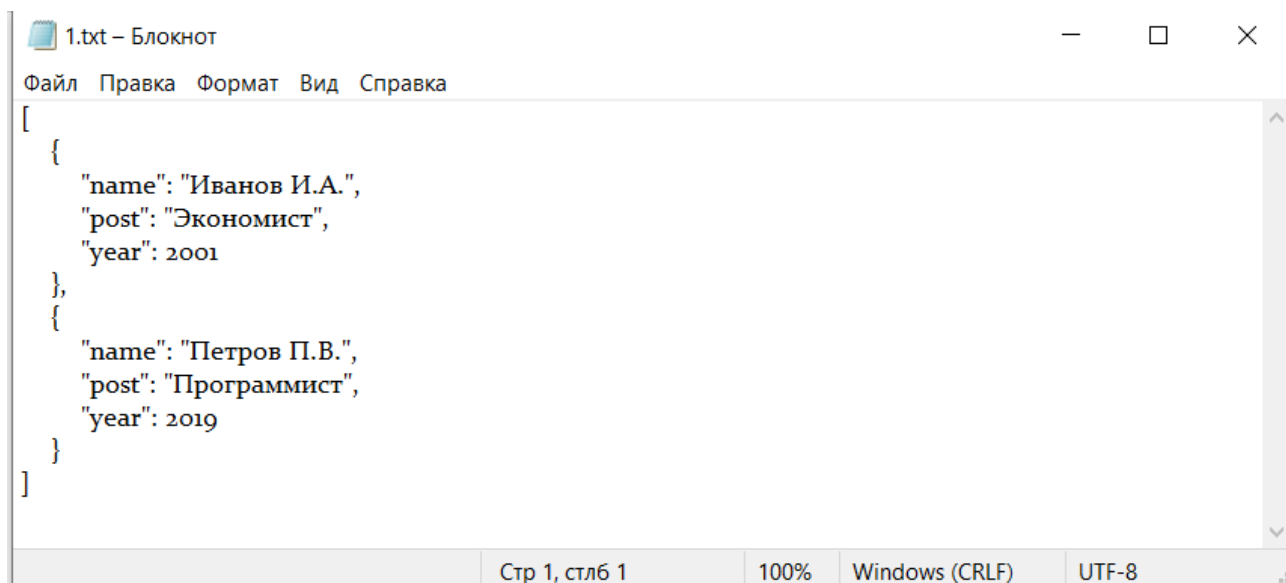
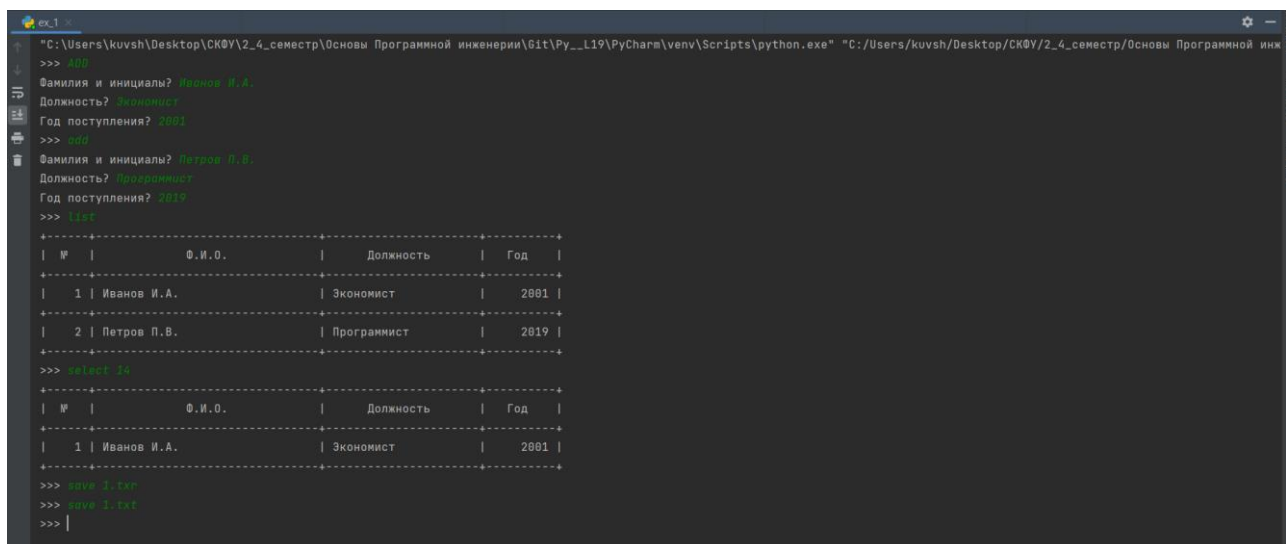


Рисунок 19.5 – Результат проработки программы

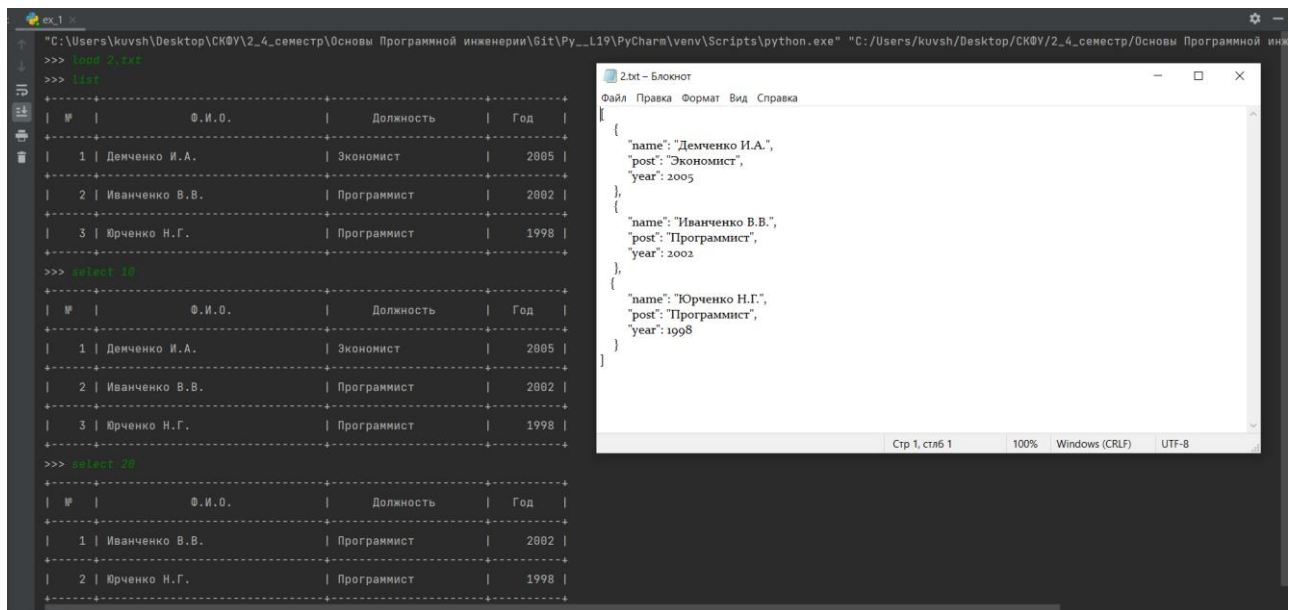


Рисунок 19.6 – Результат проработки программы

Индивидуальное 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.

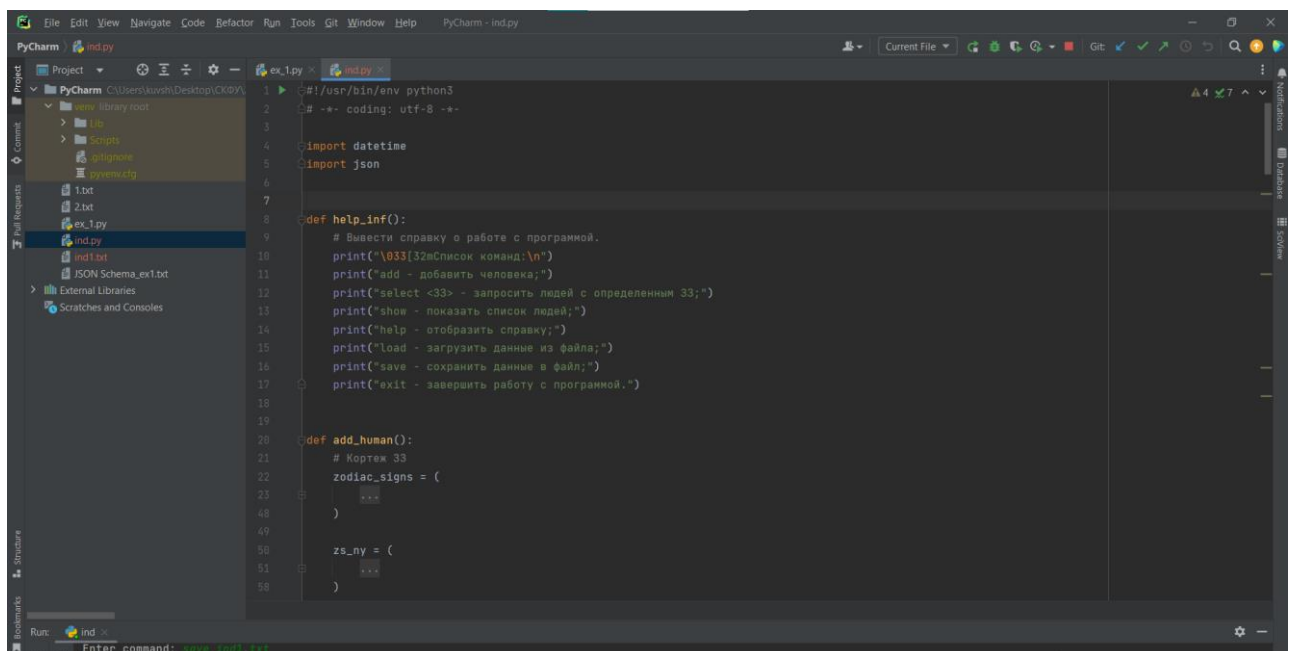


Рисунок 19.7 – Проработка индивидуального задания

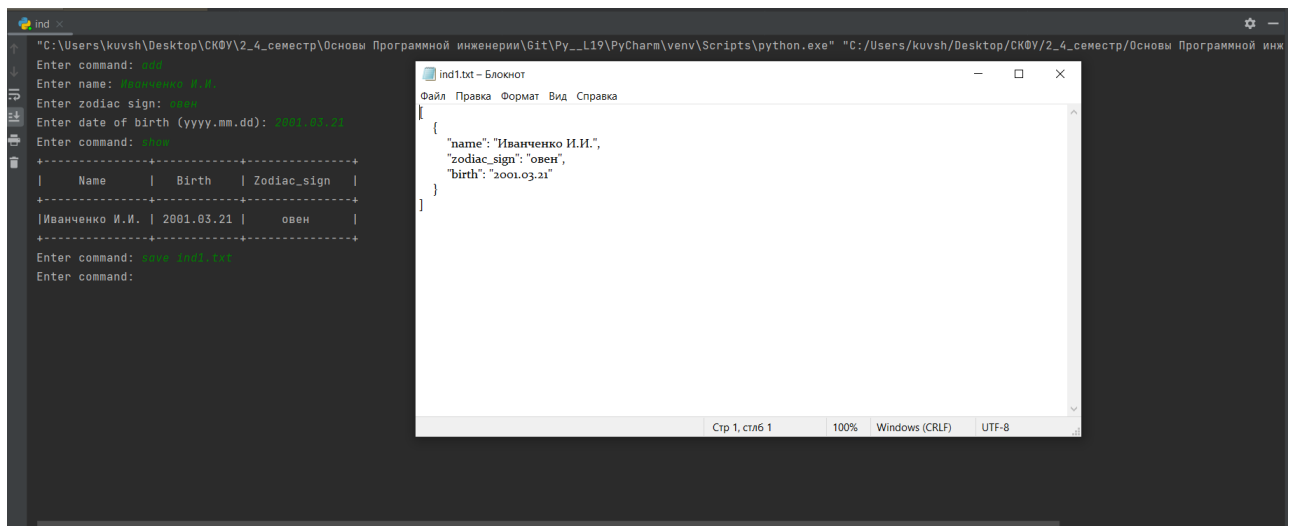


Рисунок 19.8 – Результат проработки программы

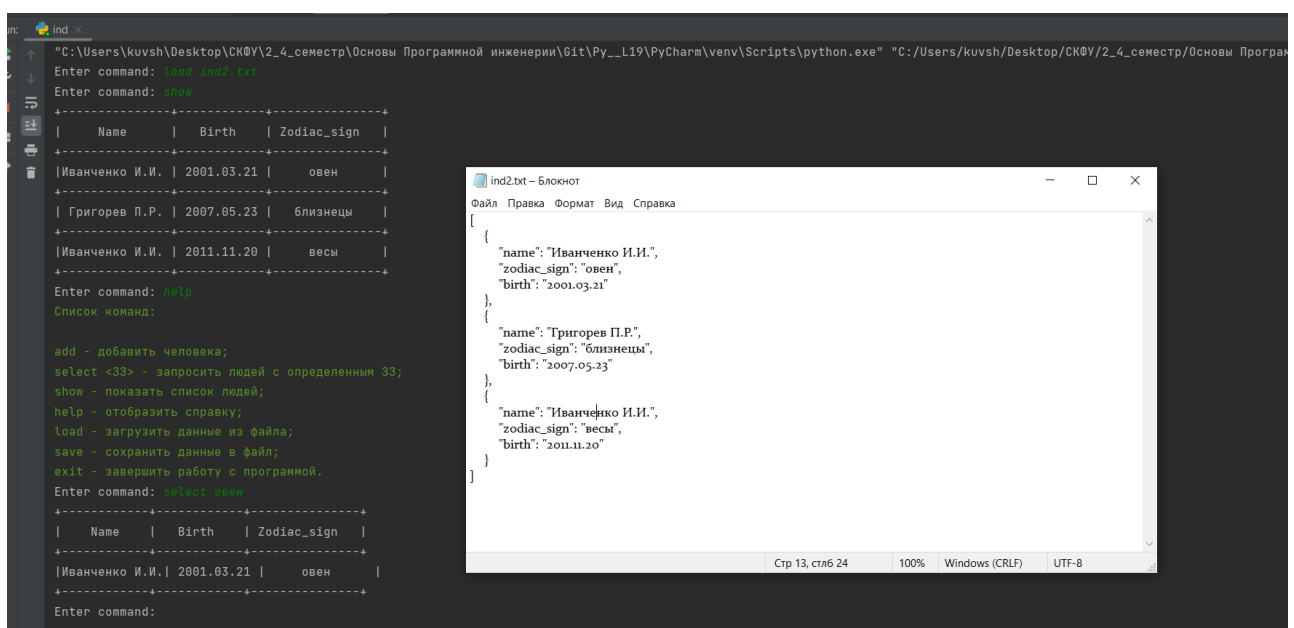


Рисунок 19.9 – Результат проработки программы

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-schema.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.

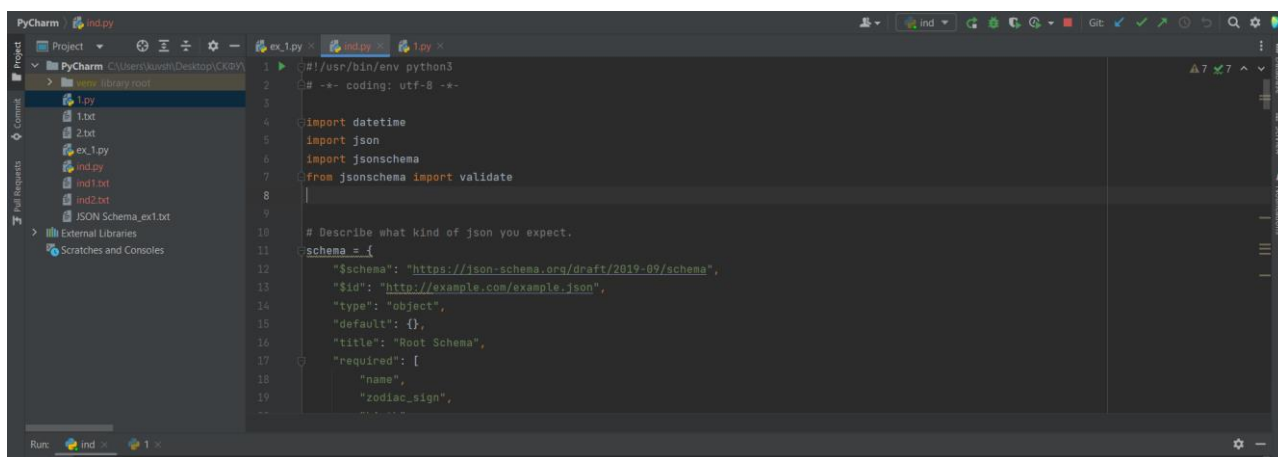


Рисунок 19.10 – Проработка индивидуального задания

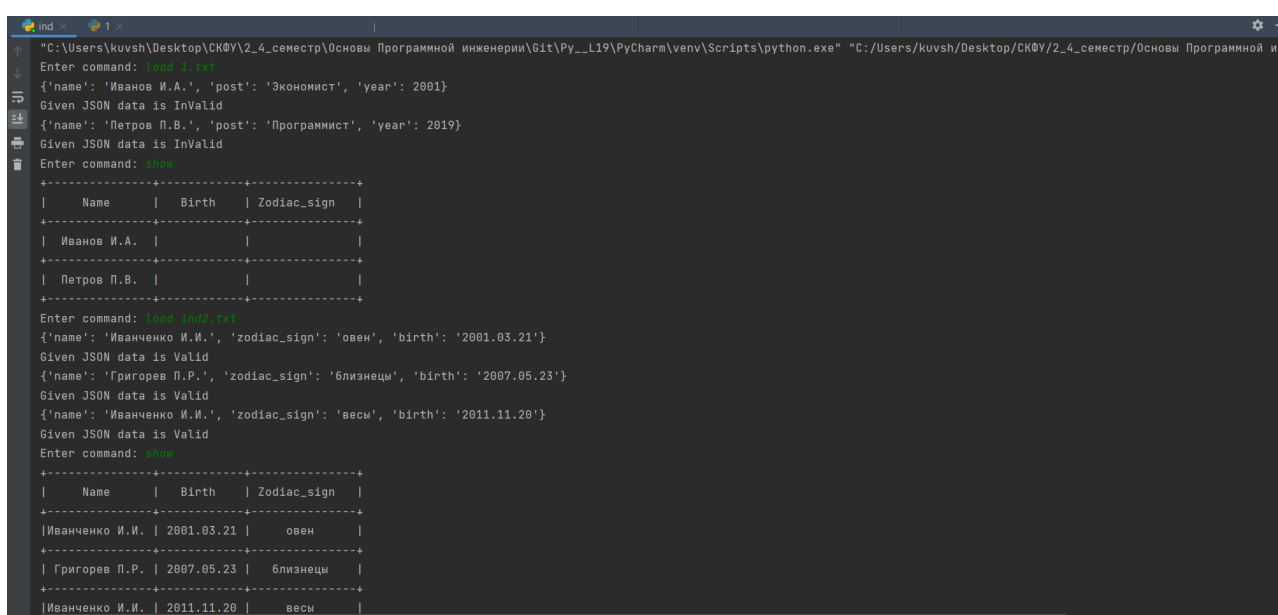


Рисунок 19.11 – Результат проработки программы

Контрольные вопросы

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как /'dʒeɪsən/ 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). Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента. Это

информативное руководство поможет вам быстрее разобраться с данными, которые вы можете использовать с JSON и основной структурой с синтаксисом этого же формата.

2. Какие типы значений используются в JSON?

Объект JSON это формат данных — ключ-значение, который обычно рендерится в фигурных скобках. Когда вы работаете с JSON, то вы скорее всего видите JSON объекты в .json файле, но они также могут быть и как JSON объект или строка уже в контексте самой программы.

Вот так выглядит JSON объект:

```
{  
  "first_name" : "sammy",  
  "last_name" : "shark",  
  "location" : "ocean",  
  "online" : true,  
  "followers" : 987  
}
```

Ключи в JSON находятся с левой стороны от двоеточия. Их нужно оборачивать в скобки, как с "key" и это может быть любая строка. В каждом объекте, ключи должны быть уникальными. Такие ключевые строки могут содержать пробелы, как в "first_name" , но такой подход может усложнить получение доступа к ним во время процесса разработки, так что лучшим вариантом в таких случаях будет использование нижнего подчеркивания, как сделано тут "first_name" . JSON значения находятся с правой стороны от двоеточия. Если быть точным, то им нужно быть одним из шести типов данных: строкой, числом, объектом, массивом, булевым значением или null .

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 escape последовательностей, 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?

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.

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.

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.

Объекты

Ключи объектов могут быть именами идентификаторов ECMAScript 5.1.

Объекты могут иметь одну запятую.

Массивы

Массивы могут иметь одну запятую.

Строки

Строки могут заключаться в одинарные кавычки.

Строки могут охватывать несколько строк, экранируя символы новой строки.

Строки могут включать в себя экранирование символов.

Числа

Числа могут быть шестнадцатеричными.

Числа могут иметь ведущую или последующую десятичную точку.

Числа могут быть Infinity, -Infinity2 и NaN.

Числа могут начинаться с явно определенного знака +.

Комментарии

Допускаются однострочные и многострочные комментарии.

Пробельные символы

Разрешены дополнительные пробельные символы.

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

[PyJSON5 — документация PyJSON5 1.6.2](#)

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

1. 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()?

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?

2. 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)
```

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

Выполнить сериализацию данных в формат JSON.

Для поддержки кириллицы установим ensure_ascii=False

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.

Схема JSON — это декларативный язык, позволяющий аннотировать и проверять документы JSON.

[jsonschema · PyPI](#)

[Понимание схемы JSON \(JSON Schema\), часть 1 \(infostart.ru\)](#)