



# Информационные ресурсы в финансовом мониторинге

---

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. ЛЕКЦИЯ 2

# Часть 1

---

JSON И SQLITE ФОРМАТЫ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

# JSON-формат

---

JSON (JavaScript Object Notation) – это язык разметки данных, применяемые в языке программирования JavaScript. Основное преимущество JSON-формата перед XML-форматом данных – это его лаконичность. Поэтому данный формат в последнее время является наиболее востребованным в вопросах обмена данными между информационными системами.

Стандарт формата JSON приведён на сайте [json.org](https://json.org) [1].

# JSON-формат

---

Синтаксис JSON-файла можно описать следующим образом, он может содержать:

Строка — это упорядоченное заключённое в двойные кавычки множество из нуля или более символов, записанных в кодировке Unicode. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\». Причём поддерживаются специальные варианты escape-последовательностей: \', \", \\, \/, \t, \n, \r, \f и \b, или стандартные варианты, записанные шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Литералы true, false и null.

Число — число в десятичной системе счисления, записанной в обычной или экспоненциальной форме. Разделитель целой и дробной части — точка.

Объект — 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.

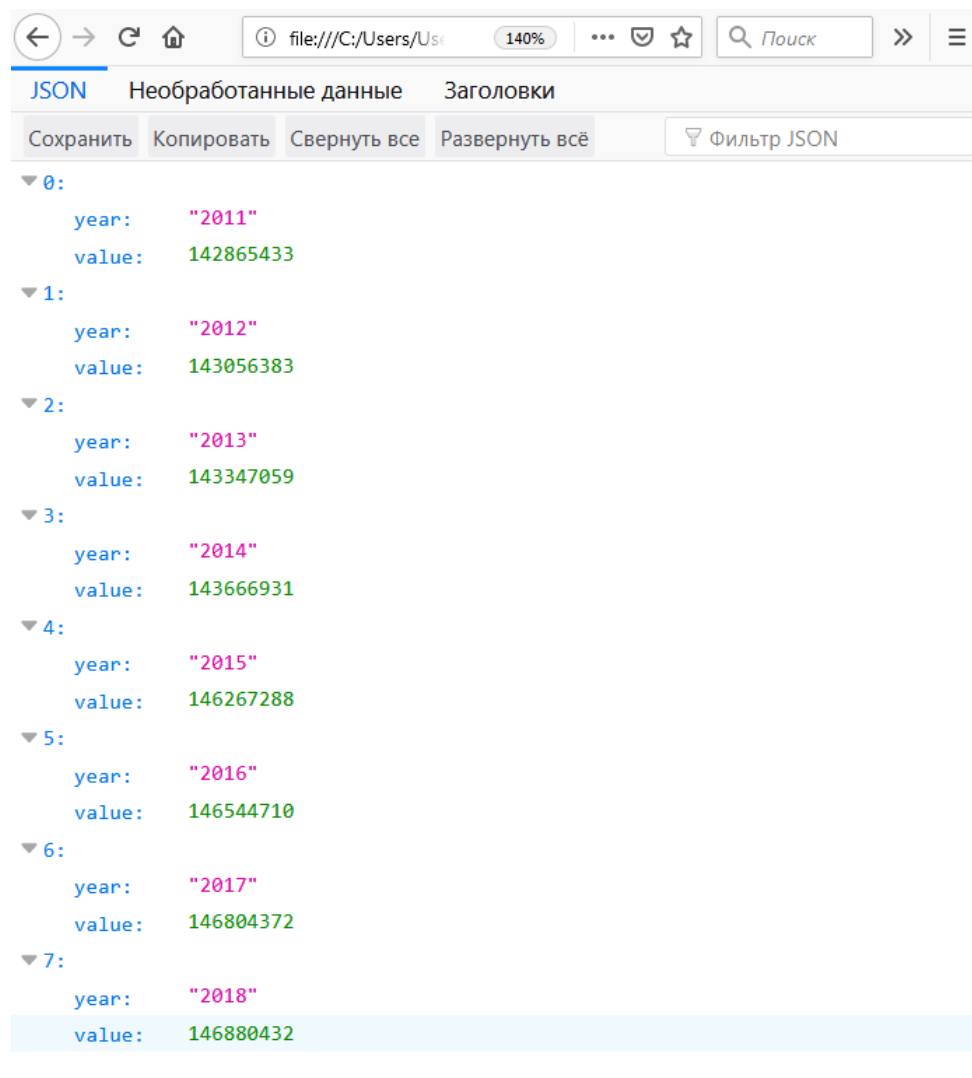
Допустима расстановка пробельных символов между всеми законченными конструкциями.

# Пример JSON-файла

---

```
[  
  {"year": "2011", "value": 142865433},  
  {"year": "2012", "value": 143056383},  
  {"year": "2013", "value": 143347059},  
  {"year": "2014", "value": 143666931},  
  {"year": "2015", "value": 146267288},  
  {"year": "2016", "value": 146544710},  
  {"year": "2017", "value": 146804372},  
  {"year": "2018", "value": 146880432},  
  {"year": "2019", "value": 146780720},  
  {"year": "2020", "value": 146748590},  
  {"year": "2021", "value": 147182123},  
  {"year": "2022", "value": 146980061},  
  {"year": "2023", "value": 146424729},  
  {"year": "2024", "value": 146150789}  
]
```

# Браузеры умеют показывать JSON



# Импорт JSON в Excel

---

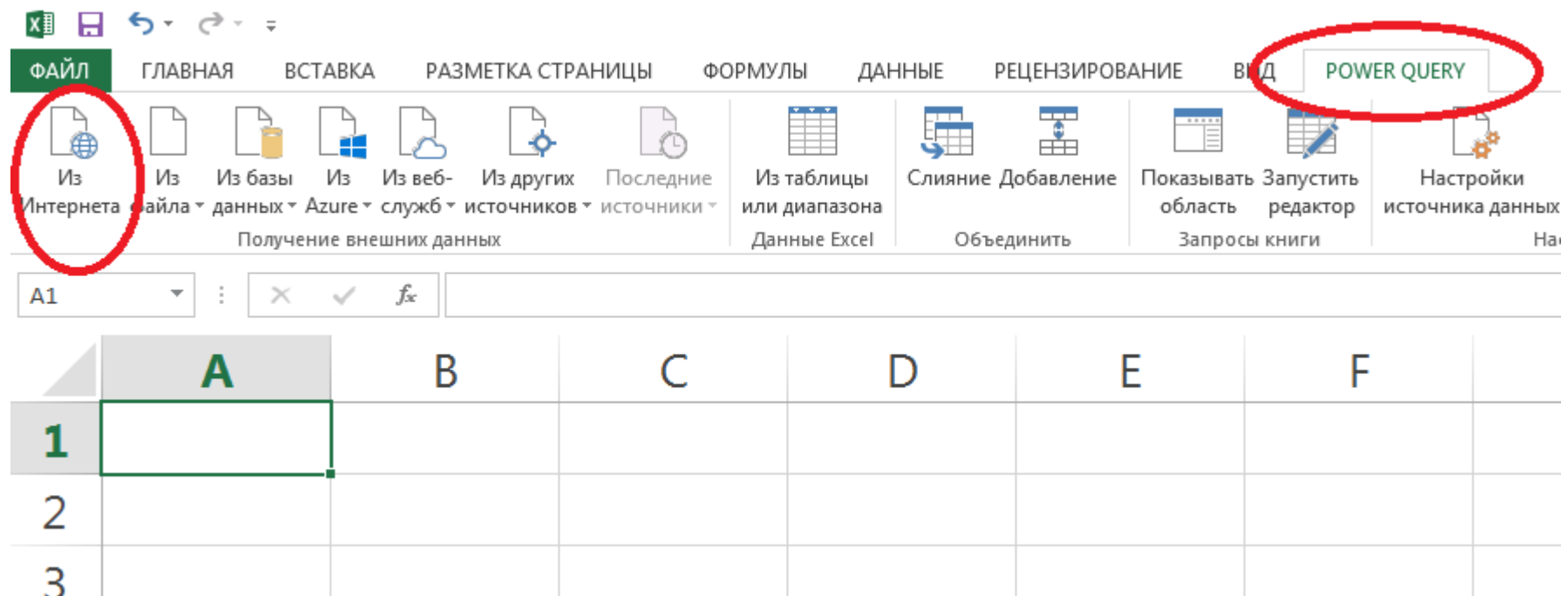
Для импорта файлов JSON в таблицу Microsoft Excel лучше всего использовать средства PowerQuery. Для MS Office 2013 данное расширение бесплатно доступно на сайте компании Microsoft [2].

После его установки в верхнем меню Microsoft Excel появляется дополнительный раздел POWER QUERY.

В более старших версиях MS Office Power Query является частью вкладки «данные».

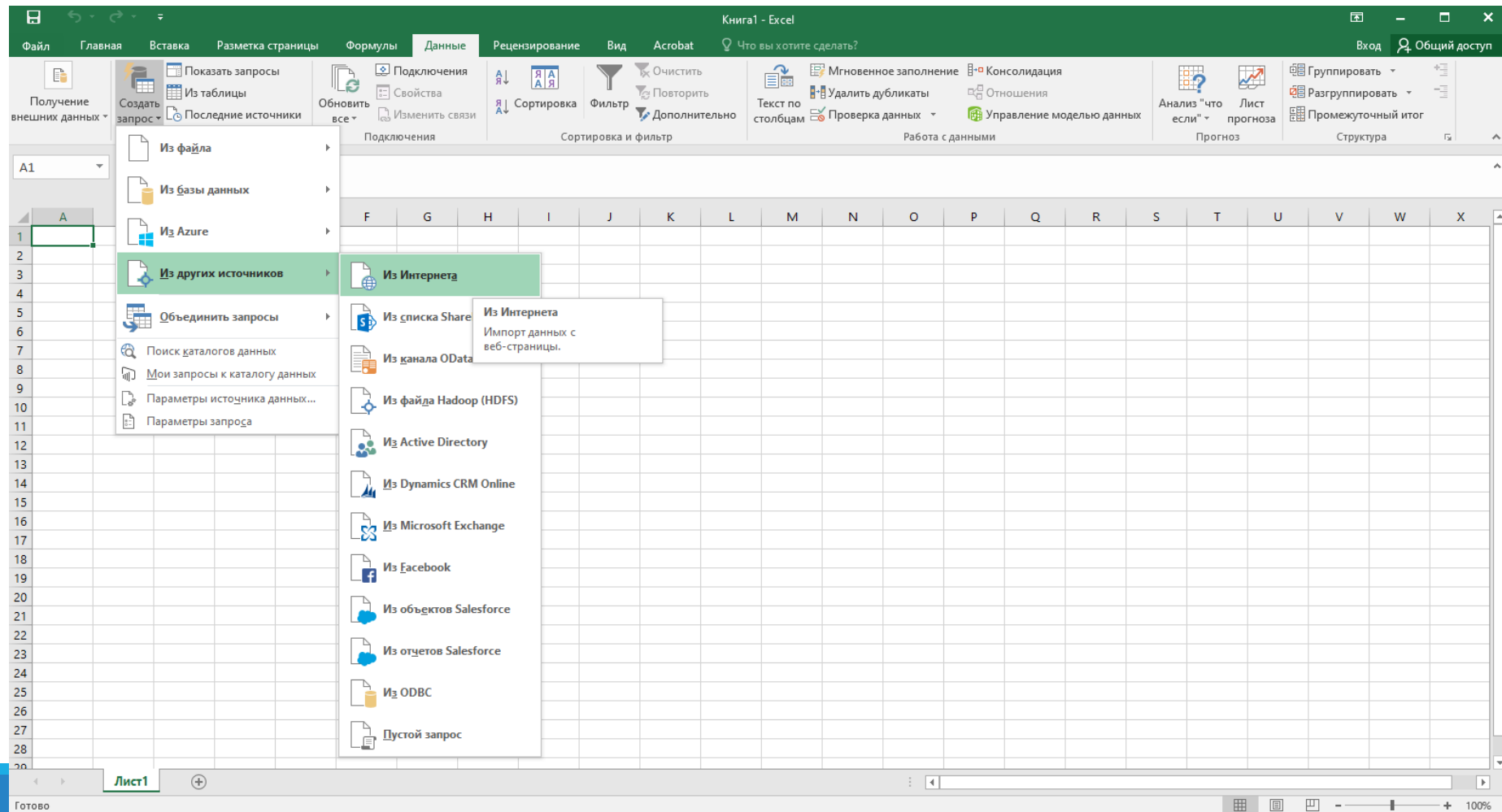
Рассмотрим, как импортировать наш файла в Microsoft Excel.

# Шаг 1. Импорт из интернета (Office 2013)

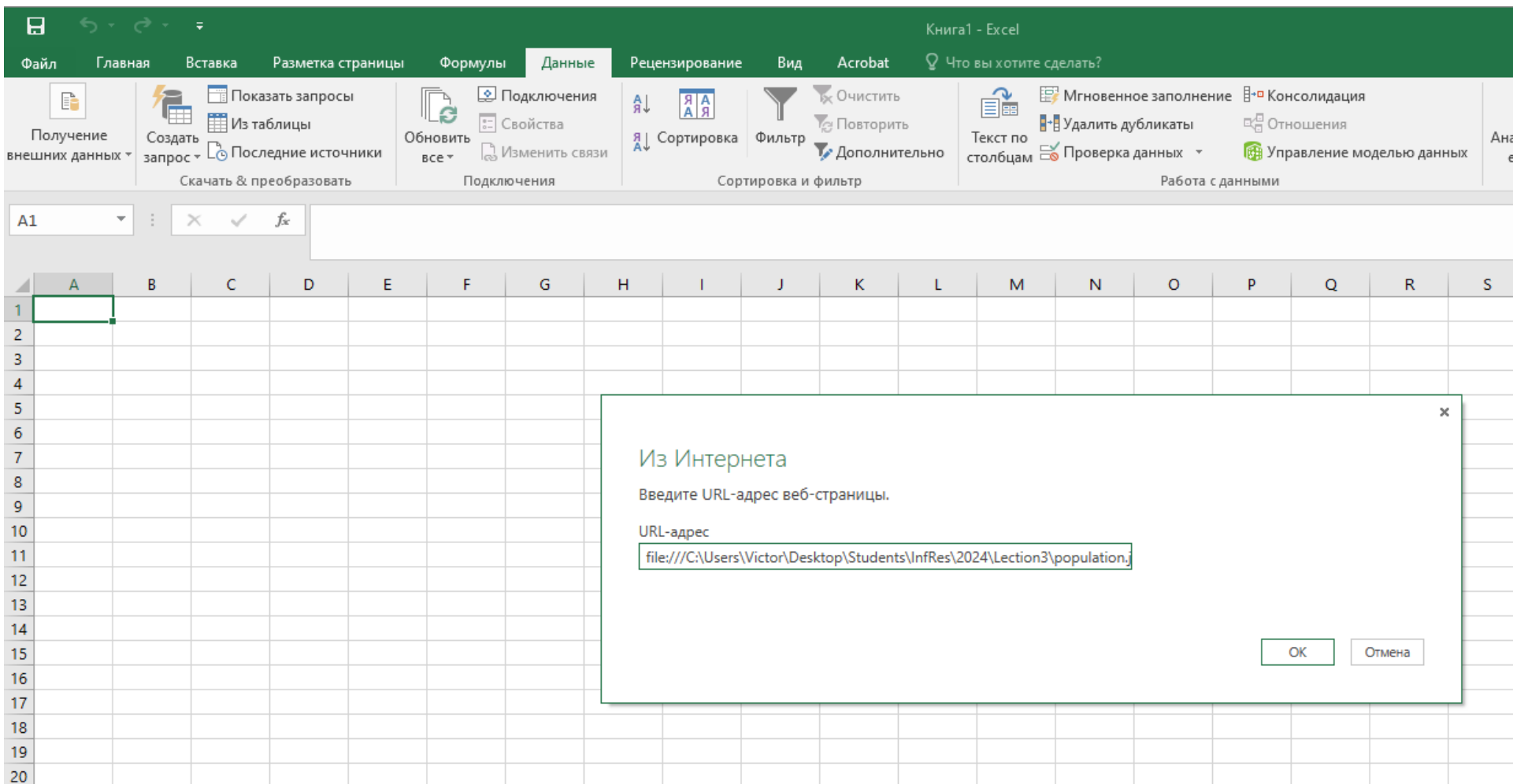




# Шаг 1. Импорт из интернета (Office 2016)



# Шаг 2. Указание папки с файлом



# Шаг 3. Преобразование в таблицу

Книга1 - Excel

Файл Главная Вставка Разметка страницы Формулы Данные Рецензирование Вид Acrobat Что вы хотите сделать? Вход Общ

Получение внешних данных

Показать запросы Из таблицы Подключения Свойства Средства для списков Запрос1 - редактор запросов

Файл Главная Преобразование Добавить столбец Просмотр Преобразовать

В таблицу Сохранить элементы Удалить элементы Удалить повторения Расположить элементы в обратном порядке Сортировать Нумерованный список

Преобразовать этот список в таблицу.

Запросы

2	Record
3	Record
4	Record
5	Record
6	Record
7	Record
8	Record
9	Record
10	Record
11	Record
12	Record
13	Record
14	Record
15	Record
16	Record
17	Record
18	
19	
20	

В таблицу

Создать таблицу из списка значений.

Выберите или введите разделитель

Нет

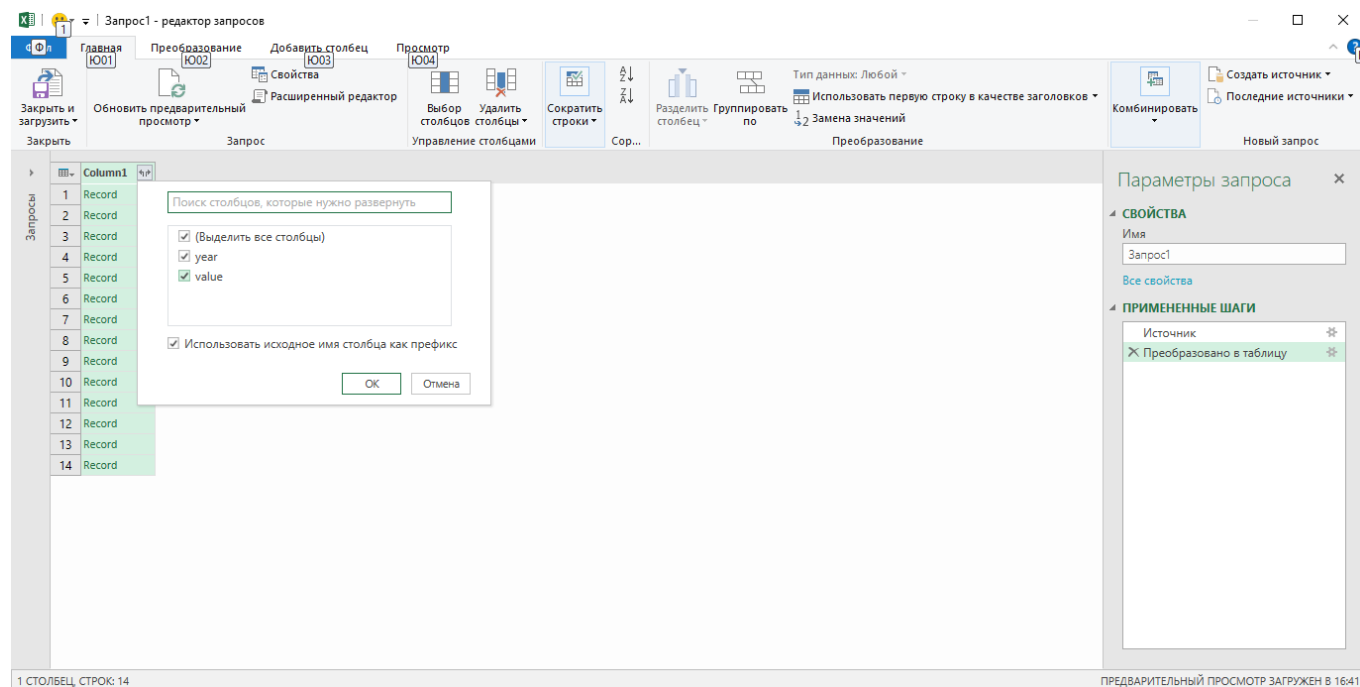
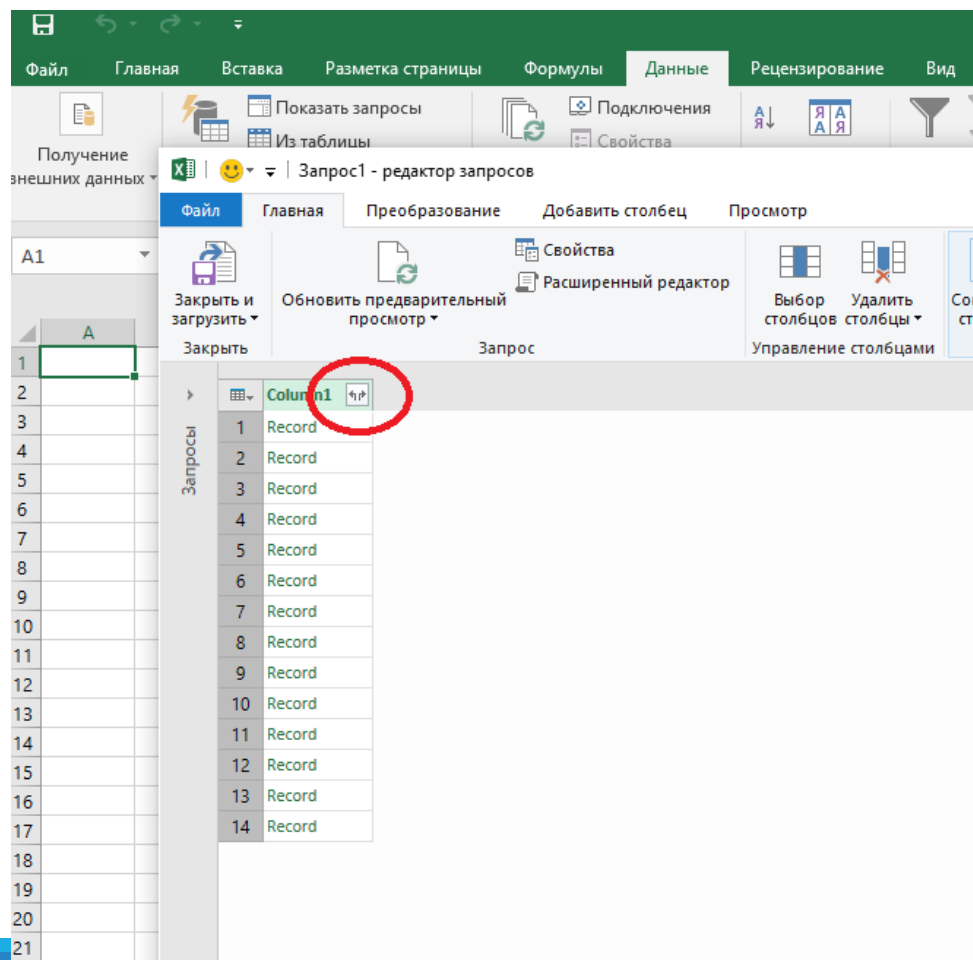
Как обрабатывать дополнительные столбцы

Показывать как ошибочные

Параметры запроса

OK Отмена

# Шаг 4. Раскрытие структуры элементов



# Шаг 5. Финальная загрузка

Запрос1 - редактор запросов

Главная | Преобразование | Добавить столбец | Просмотр

Закрыть и загрузить | Обновить предварительный просмотр | Свойства | Расширенный редактор | Выбор столбцов | Удалить столбцы | Сократить строки | Разделить столбец | Группировать по | Тип данных: Любой | Использовать первую строку в качестве заголовков | Комбинировать | Создать источник | Последние источники | Новый запрос

	Column1.year	Column1.value
1	2011	142865433
2	2012	143056383
3	2013	143347059
4	2014	143666931
5	2015	146267288
6	2016	146544710
7	2017	146804372
8	2018	146880432
9	2019	146780720
10	2020	146748590
11	2021	147182123
12	2022	146980061
13	2023	146424729
14	2024	146150789

Параметры запроса

СВОЙСТВА

Имя: Запрос1

Все свойства

ПРИМЕНЕННЫЕ ШАГИ

- Источник
- Преобразовано в таблицу
- РАЗВЕРНУТЫЙ ЭЛЕМЕНТ Column1

СТОЛБЦОВ: 2, СТРОК: 14

ПРЕДВАРИТЕЛЬНЫЙ ПРОСМОТР ЗАГРУЖЕН В 16:43

# Результат

Книга1 - Excel

Работа с таблицами | Работа с запросами

Файл | Главная | Вставка | Разметка страницы | Формулы | Данные | Рецензирование | Вид | Acrobat | Конструктор | Запрос

Имя таблицы: Запрос1

Свойства | Сервис

Сводная таблица | Удалить дубликаты | Преобразовать в диапазон | Вставить срез | Экспорт | Обновить | Открыть в браузере | Разорвать связь | Данные из внешней таблицы

Строка заголовка | Строка итогов | Чередующиеся строки | Первый столбец | Последний столбец | Чередующиеся столбцы | Кнопка фильтра

Стили таблиц

Column1.year

Column1.year	Column1.value
2011	142865433
2012	143056383
2013	143347059
2014	143666931
2015	146267288
2016	146544710
2017	146804372
2018	146880432
2019	146780720
2020	146748590
2021	147182123
2022	146980061
2023	146424729
2024	146150789

Запросы книги

1 запрос

Запрос1

Загружено строк: 14.

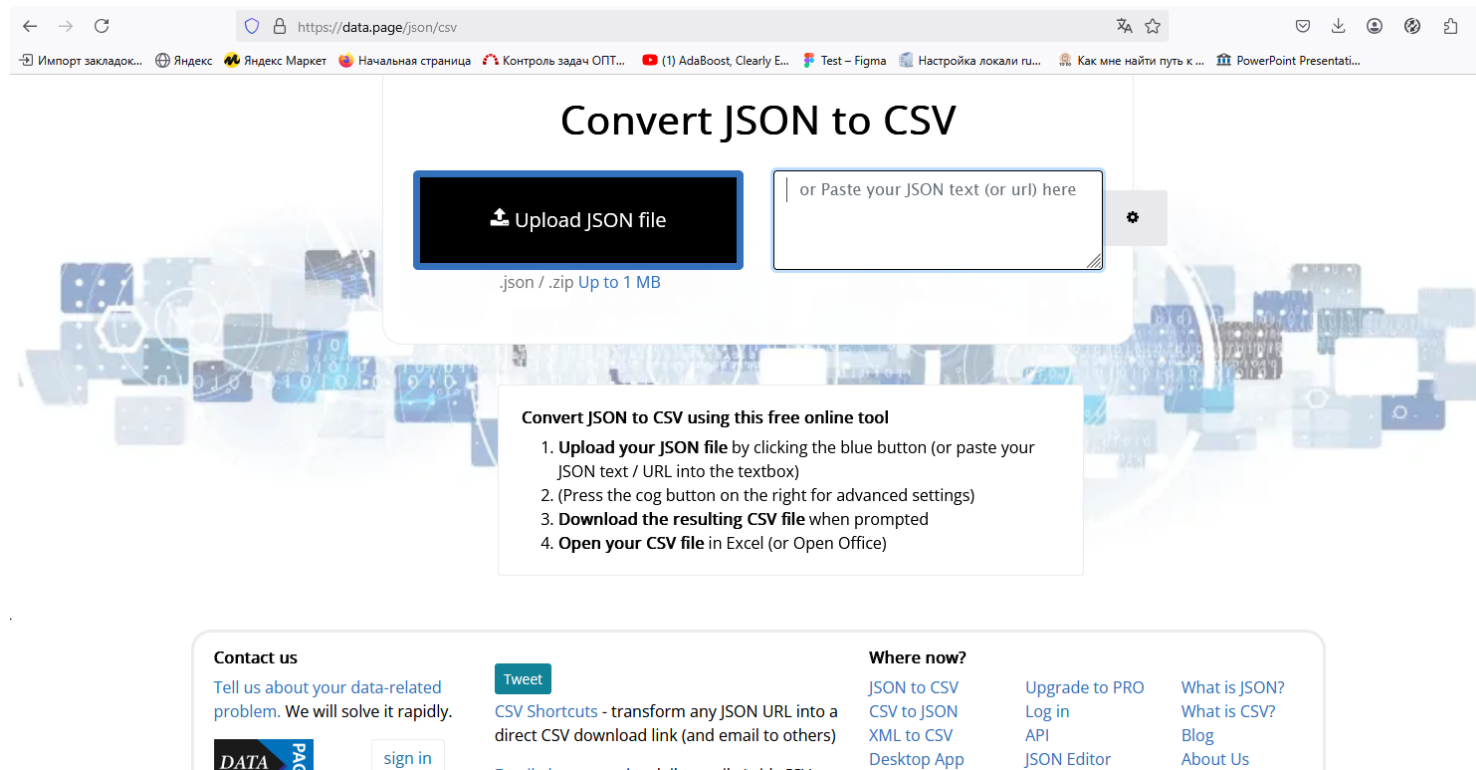
Лист1 | Лист2

Готово

Среднее: 145692830 | Количество: 30 | Максимум: 147182123 | Сумма: 2039699620

# Онлайн-сервисы

Если ваша информация не является секретной, то можно выполнить преобразование JSON-файла в другие форматы посредством онлайн сервисов. Например, для преобразования JSON в CSV можно использовать сайт <https://json-csv.com/>.



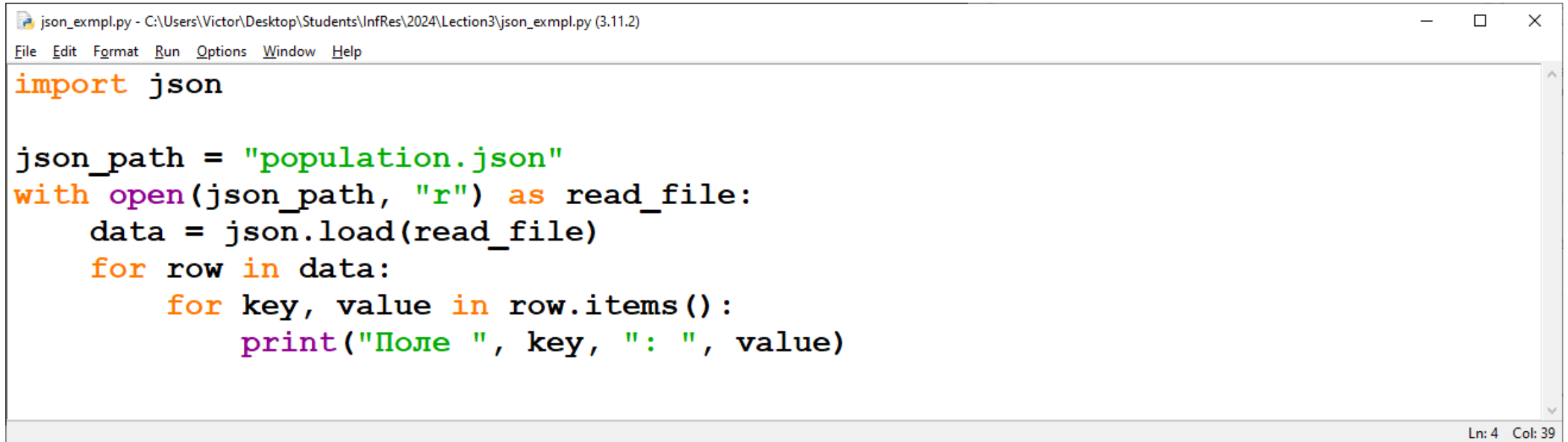
# json для Python

---

Библиотека `json` встроена в стандартную поставку языка Python и позволяет выполнять как чтение и разбор файлов в данном формате, так и их запись.



# Пример чтения и разбора JSON-файла



The screenshot shows a Python IDE window titled "json\_exmpl.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\json\_exmpl.py (3.11.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code in the editor is as follows:

```
import json

json_path = "population.json"
with open(json_path, "r") as read_file:
    data = json.load(read_file)
    for row in data:
        for key, value in row.items():
            print("Поле ", key, ": ", value)
```

The status bar at the bottom right indicates "Ln: 4 Col: 39".

# Пример чтения и разбора JSON-файла

---

```
import json

json_path = "population.json"
with open(json_path, "r") as read_file:
    data = json.load(read_file)
    for row in data:
        for key, value in row.items():
            print("Поле ", key, ": ", value)
```

# Результат

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
>>>
= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\json_exmpl.py =
Поле year : 2011
Поле value : 142865433
Поле year : 2012
Поле value : 143056383
Поле year : 2013
Поле value : 143347059
Поле year : 2014
Поле value : 143666931
Поле year : 2015
Поле value : 146267288
Поле year : 2016
Поле value : 146544710
Поле year : 2017
Поле value : 146804372
Поле year : 2018
Поле value : 146880432
Поле year : 2019
Поле value : 146780720
Поле year : 2020
Поле value : 146748590
Поле year : 2021
Поле value : 147182123
Поле year : 2022
Поле value : 146980061
Поле year : 2023
Поле value : 146424729
Поле year : 2024
Поле value : 146150789
>>>
```

# СУБД SQLite

---

SQLite – это не СУБД в стандартном её понимании, а библиотека, позволяющая встроить в приложение механизм работы с СУБД. То есть, сам по себе SQLite не существует. Но если Вы пишете свою программу, то можете подключить библиотеку SQLite, и из своей программы средствами этой библиотеки создать и использовать репозиторий базы данных. Говоря языком википедии, SQLite – это встраиваемая СУБД.

Весь репозиторий базы данных (вся база) хранится в одном единственном файле.

# Работа с СУБД из языка высокого уровня

---

Работа с СУБД из языка высокого уровня, например, Java, Ruby, Python, C++ и т.д. осуществляется в простейшем случае (когда не используются фреймворки с ORM-модулями) с помощью специальных библиотек. Библиотеки бывают низкоуровневыми и высокоуровневыми.

Низкоуровневые библиотеки ориентированы на конкретную СУБД (например, Oracle) и не позволяют работать с продуктами других производителей. Такие библиотеки называют драйвером СУБД.

Высокоуровневые библиотеки являются универсальными средствами, способными в одном виде работать с любыми СУБД. В этом случае, помимо библиотеки, требуется подключить драйвер конкретной(ых) СУБД.

Фактически, данные библиотеки отправляют написанный в виде строки SQL-запрос по сети к СУБД, получают от неё ответ и позволяют данный ответ преобразовать в вид, подходящий для языка высокого уровня.

# Работа с СУБД из языка высокого уровня

---

1. Установка соединения с СУБД.
2. Формирование SQL-утверждения.
3. Выполнение SQL-утверждения.
4. Построчный (или полный) сбор результатов.
5. Закрытие соединения с СУБД.

\* В некоторых библиотеках также требуется закрывать SQL-утверждение.

# sqlite3 для Python

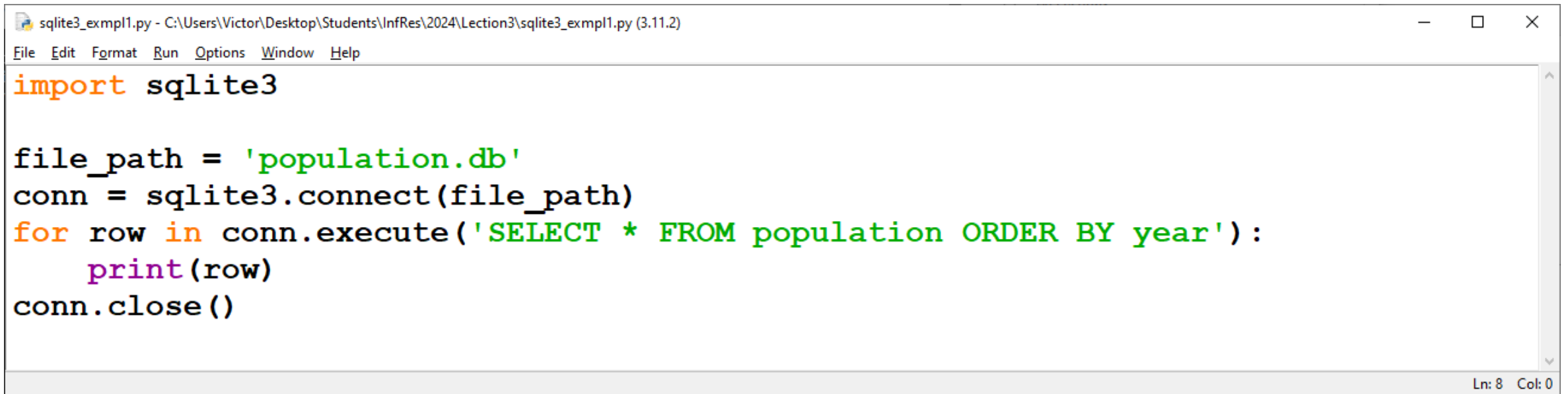
---

Библиотека sqlite3 встроена в стандартную поставку языка Python и позволяет выполнять как команды DML, так и команды DDL с СУБД SQLite.

1. Установка соединения:  
`дескриптор_соединения = sqlite3.connect`
2. Формирование SQL-утверждения:  
`дескриптор_курсора = дескриптор_соединения.cursor()`
3. Выполнение SQL-утверждения: `дескриптор_соединения.execute(sql, параметры)` или `дескриптор_курсора.execute(sql, параметры)`
4. Построчный (или полный) сбор результатов:  
итерацией по результату execute `for row in результат_execute`  
загрузка одной строки из курсора `дескриптор_курсора.fetchone()`  
загрузка всех строк из курсора `дескриптор_курсора.fetchall()`
5. Закрытие соединения с СУБД:  
`дескриптор_соединения.close()`

# Простейший пример

---



The screenshot shows a Python IDE window titled "sqlite3\_exmpl1.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lecton3\sqlite3\_exmpl1.py (3.11.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code in the editor is as follows:

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
for row in conn.execute('SELECT * FROM population ORDER BY year'):
    print(row)
conn.close()
```

The status bar at the bottom right indicates "Ln: 8 Col: 0".



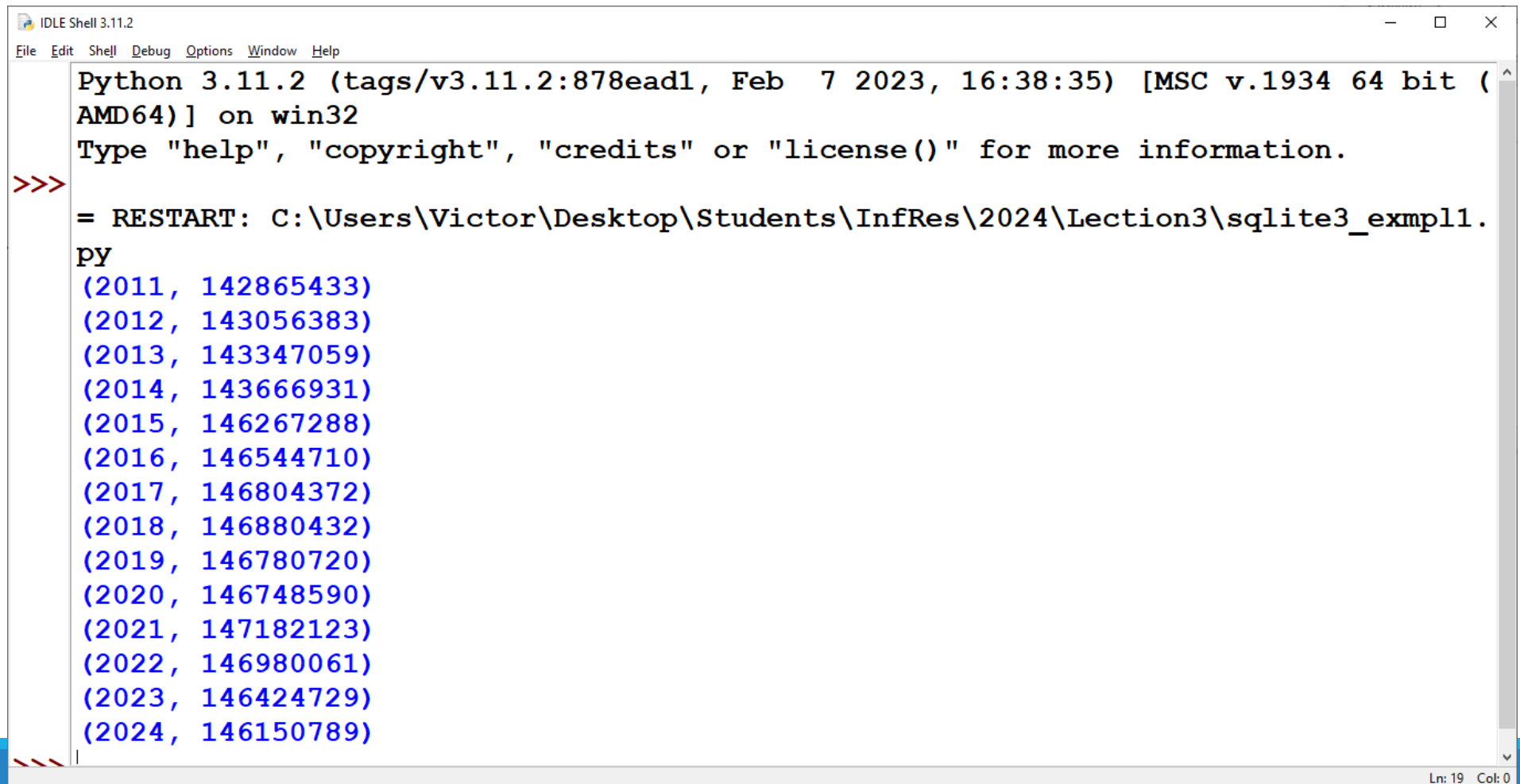
# Простейший пример

---

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
for row in conn.execute('SELECT * FROM population ORDER BY year'):
    print(row)
conn.close()
```

# Результат

A screenshot of a Python IDLE Shell window. The window title is 'IDLE Shell 3.11.2'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell displays the following text: 'Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', and a red prompt '>>>'. Below the prompt, it shows '= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3\_exmpl1.py'. This is followed by a list of 12 tuples, each containing a year and a number, displayed in blue text: (2011, 142865433), (2012, 143056383), (2013, 143347059), (2014, 143666931), (2015, 146267288), (2016, 146544710), (2017, 146804372), (2018, 146880432), (2019, 146780720), (2020, 146748590), (2021, 147182123), (2022, 146980061), (2023, 146424729), and (2024, 146150789). The status bar at the bottom right shows 'Ln: 19 Col: 0'.

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3_exmpl1.py
(2011, 142865433)
(2012, 143056383)
(2013, 143347059)
(2014, 143666931)
(2015, 146267288)
(2016, 146544710)
(2017, 146804372)
(2018, 146880432)
(2019, 146780720)
(2020, 146748590)
(2021, 147182123)
(2022, 146980061)
(2023, 146424729)
(2024, 146150789)
Ln: 19 Col: 0
```

# Добавление строки

```
sqlite3_exmpl2.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3_exmpl2.py (3.11.2)
File Edit Format Run Options Window Help

import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
sql = "INSERT INTO population VALUES (?, ?)"
cursor = conn.cursor()
cursor.execute(sql, [2025, 147000000])
conn.commit()
for row in cursor.execute('SELECT * FROM population ORDER BY year'):
    print(row)
conn.close()
```

Ln: 1 Col: 0

# Добавление строки

---

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
sql = "INSERT INTO population VALUES(?, ?)"
cursor = conn.cursor()
cursor.execute(sql, [2025, 147000000])
conn.commit()

for row in cursor.execute('SELECT * FROM population ORDER BY year'):
    print(row)

conn.close()
```

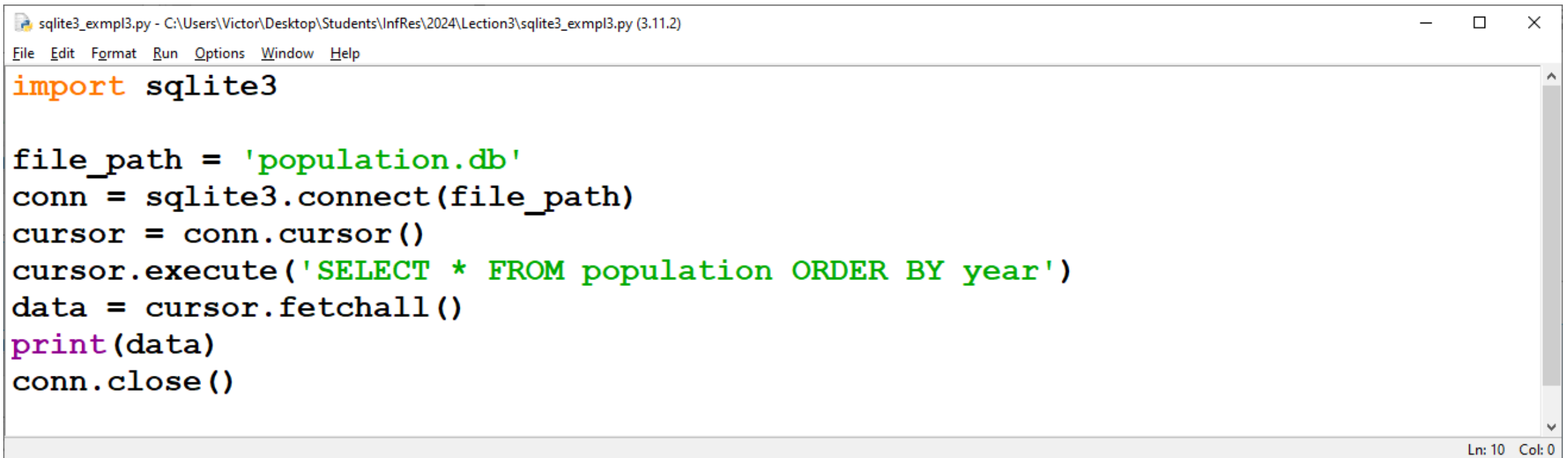
# Результат



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3_exmpl2.py
(2011, 142865433)
(2012, 143056383)
(2013, 143347059)
(2014, 143666931)
(2015, 146267288)
(2016, 146544710)
(2017, 146804372)
(2018, 146880432)
(2019, 146780720)
(2020, 146748590)
(2021, 147182123)
(2022, 146980061)
(2023, 146424729)
(2024, 146150789)
(2025, 147000000)
>>>
```

# Выбор всех строк сразу

---



```
sqlite3_exmpl3.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lec3\sqlite3_exmpl3.py (3.11.2)
File Edit Format Run Options Window Help

import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
cursor = conn.cursor()
cursor.execute('SELECT * FROM population ORDER BY year')
data = cursor.fetchall()
print(data)
conn.close()
```

Ln: 10 Col: 0

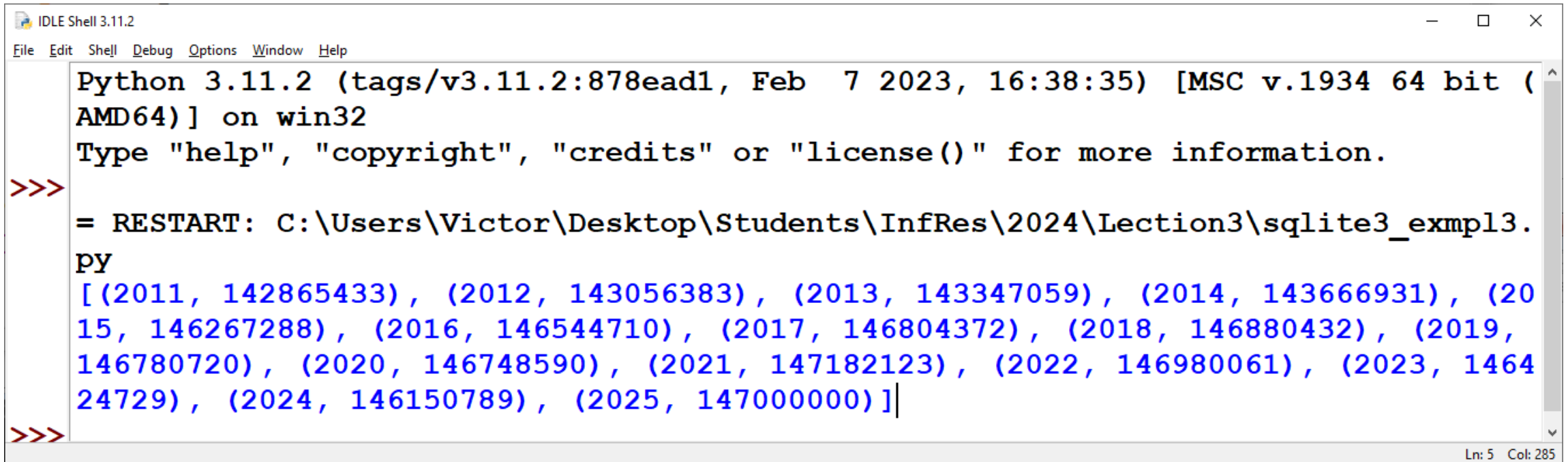
# Выбор всех строк сразу

---

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
cursor = conn.cursor()
cursor.execute('SELECT * FROM population ORDER BY year')
data = cursor.fetchall()
print(data)
conn.close()
```

# Результат



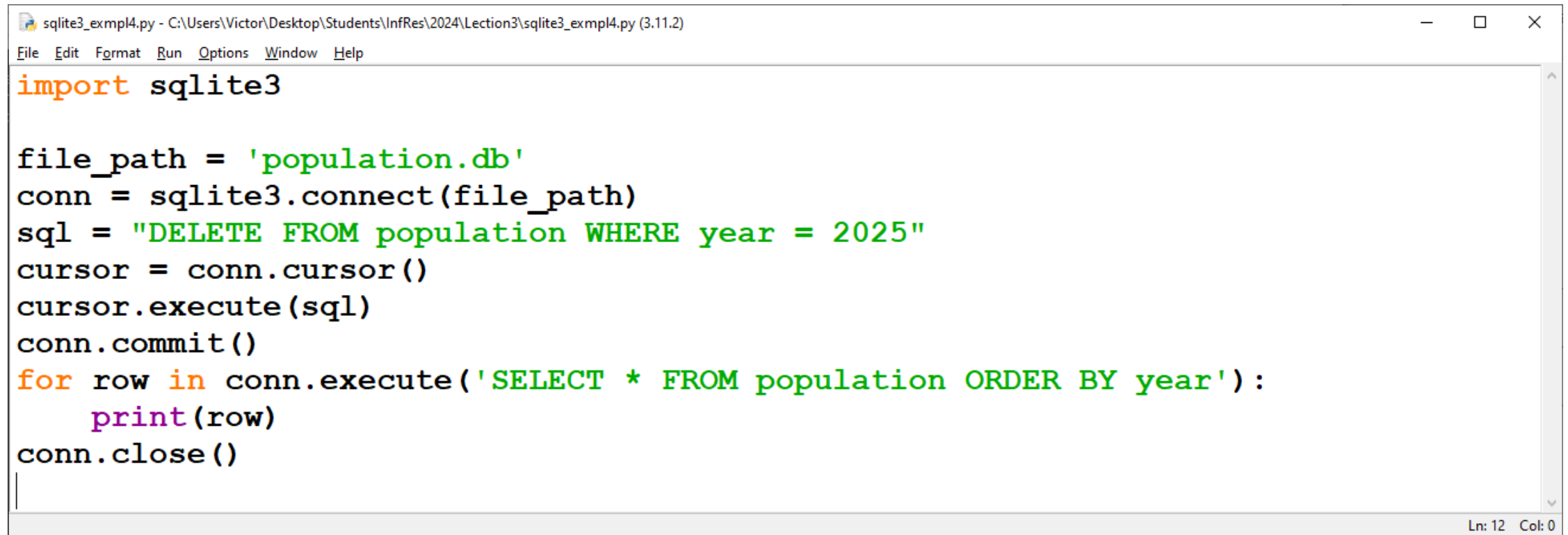
```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3_exmpl3.py
[(2011, 142865433), (2012, 143056383), (2013, 143347059), (2014, 143666931), (2015, 146267288), (2016, 146544710), (2017, 146804372), (2018, 146880432), (2019, 146780720), (2020, 146748590), (2021, 147182123), (2022, 146980061), (2023, 146424729), (2024, 146150789), (2025, 147000000)]
>>>
```

Ln: 5 Col: 285



# Удаление строки

---



The screenshot shows a Python IDE window titled "sqlite3\_exmpl4.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3\_exmpl4.py (3.11.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code in the editor is as follows:

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
sql = "DELETE FROM population WHERE year = 2025"
cursor = conn.cursor()
cursor.execute(sql)
conn.commit()
for row in conn.execute('SELECT * FROM population ORDER BY year'):
    print(row)
conn.close()
```

The status bar at the bottom right indicates "Ln: 12 Col: 0".

# Удаление строки

---

```
import sqlite3

file_path = 'population.db'
conn = sqlite3.connect(file_path)
sql = "DELETE FROM population WHERE year = 2025"
cursor = conn.cursor()
cursor.execute(sql)
conn.commit()

for row in conn.execute('SELECT * FROM population ORDER BY year'):
    print(row)
conn.close()
```

# Результат



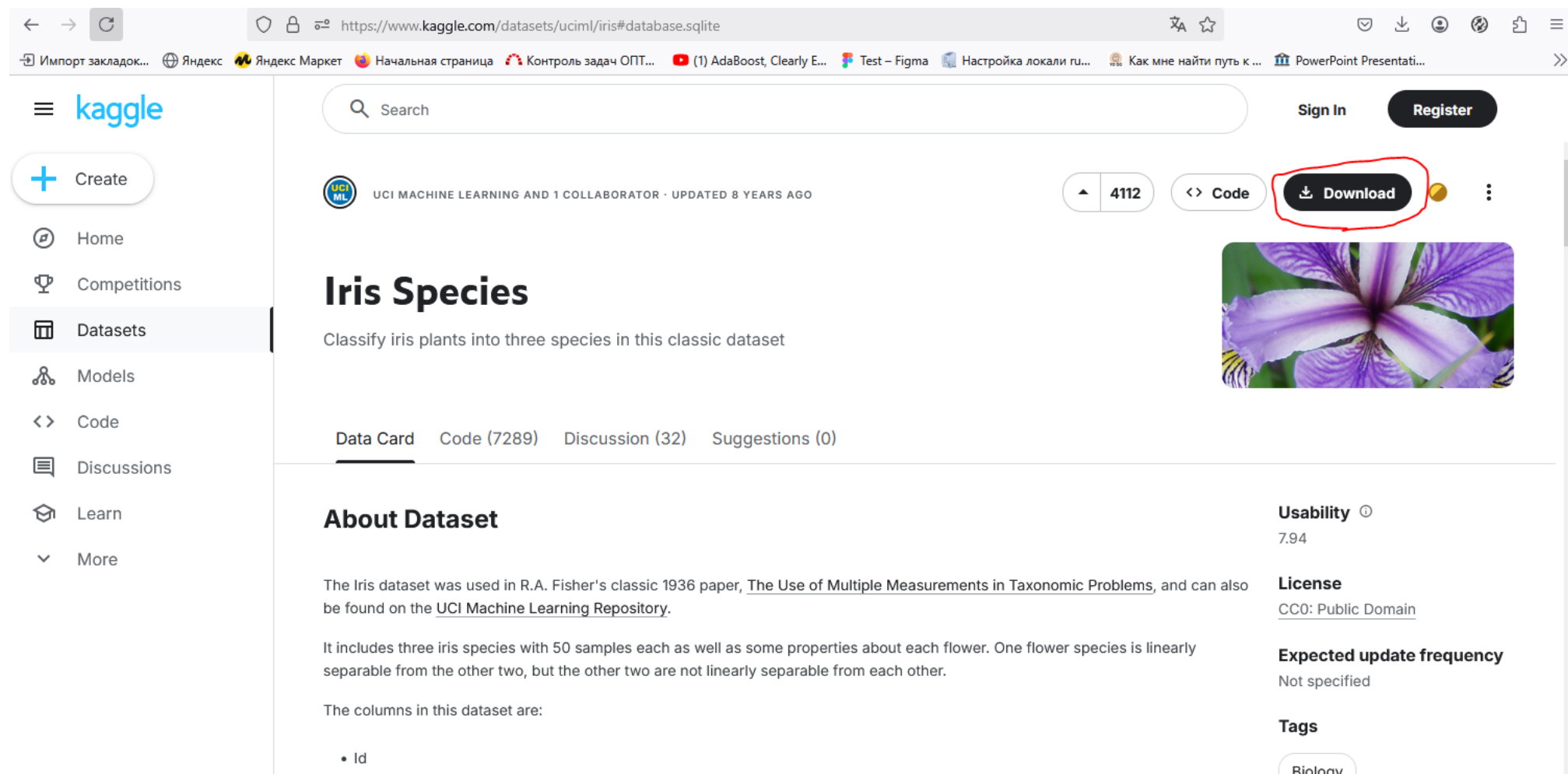
```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Victor\Desktop\Students\InfRes\2024\Lecture3\sqlite3_exmpl4.py
(2011, 142865433)
(2012, 143056383)
(2013, 143347059)
(2014, 143666931)
(2015, 146267288)
(2016, 146544710)
(2017, 146804372)
(2018, 146880432)
(2019, 146780720)
(2020, 146748590)
(2021, 147182123)
(2022, 146980061)
(2023, 146424729)
(2024, 146150789)
>>>
```

# Пример решения простейшей задачи

---

Загрузите с сайта <https://www.kaggle.com> набор данных со спецификацией ирисов [5] в формате SQLite. Преобразуйте загруженный набор данных в JSON-формат.

# Шаг 1. Загрузка набора данных



The screenshot shows the Kaggle website interface. The browser address bar displays the URL: <https://www.kaggle.com/datasets/uciml/iris#database.sqlite>. The left sidebar contains navigation links: Home, Competitions, Datasets (selected), Models, Code, Discussions, Learn, and More. The main content area features the 'Iris Species' dataset page. At the top, there is a search bar, 'Sign In', and 'Register' buttons. Below the dataset title, there are buttons for '4112', 'Code', and 'Download' (highlighted with a red circle). A red arrow points from the 'Download' button to the 'About Dataset' section. The 'About Dataset' section includes a description of the dataset, its history, and the columns it contains. The 'Usability' score is 7.94, and the license is CC0: Public Domain. The 'Expected update frequency' is 'Not specified'. The 'Tags' section shows 'Biology'.

**Iris Species**  
Classify iris plants into three species in this classic dataset

4112 Code **Download**

**About Dataset**

The Iris dataset was used in R.A. Fisher's classic 1936 paper, [The Use of Multiple Measurements in Taxonomic Problems](#), and can also be found on the [UCI Machine Learning Repository](#).

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id

**Usability** 7.94

**License** CC0: Public Domain

**Expected update frequency** Not specified

**Tags** Biology

## Шаг 2. Узнаем названия таблиц

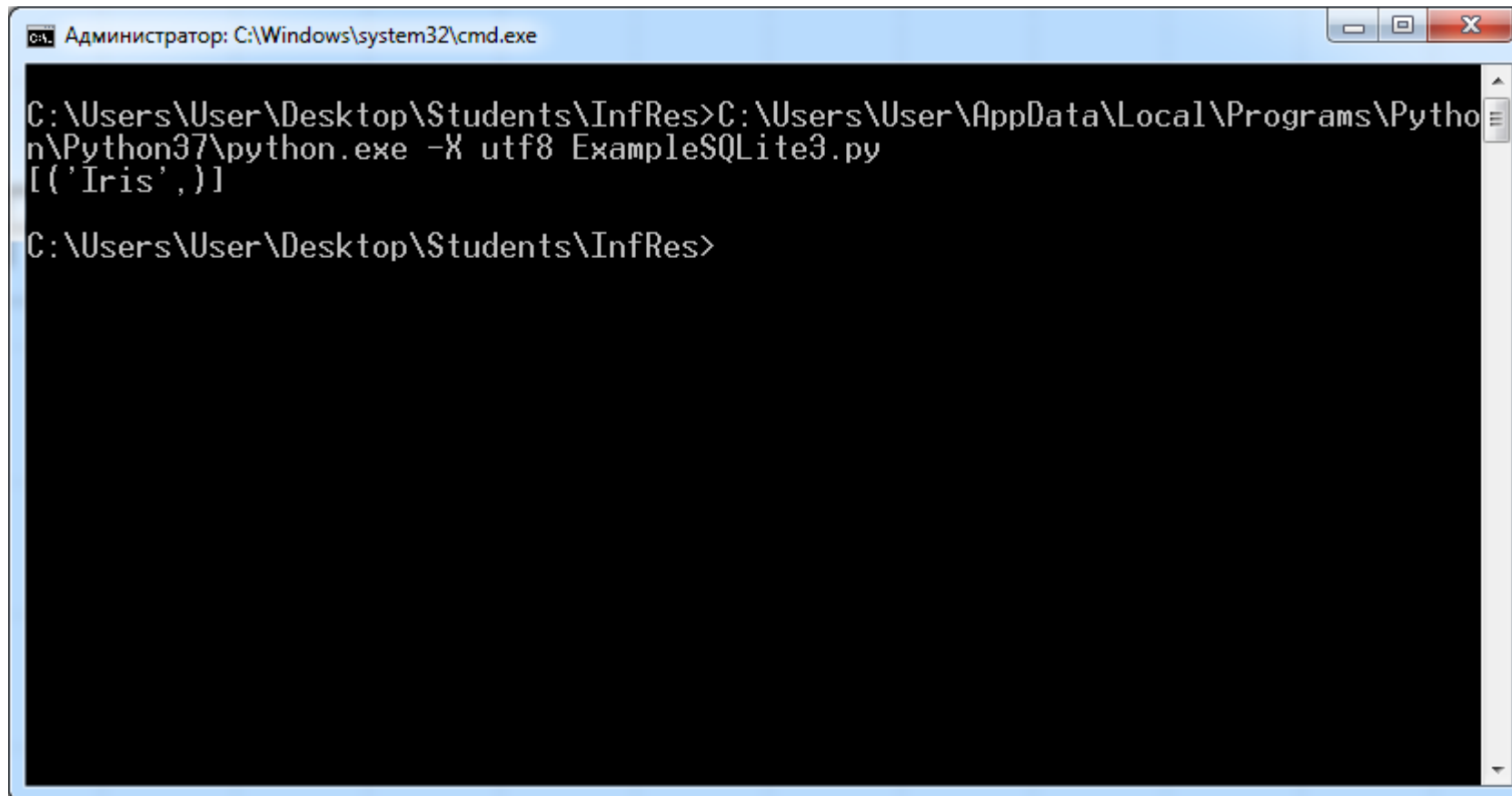
---

```
import sqlite3

file_path = 'database.sqlite'
conn = sqlite3.connect(file_path)
cursor = conn.cursor()
cursor.execute("SELECT name FROM sqlite_master WHERE type = 'table'")
data = cursor.fetchall()
print(data)
conn.close()
```

# Результат

---



The screenshot shows a Windows command prompt window titled "Администратор: C:\Windows\system32\cmd.exe". The command prompt displays the following text:

```
C:\Users\User\Desktop\Students\InfRes>C:\Users\User\AppData\Local\Programs\Python\Python37\python.exe -X utf8 ExampleSQLite3.py  
[('Iris',)]  
C:\Users\User\Desktop\Students\InfRes>
```

## Шаг 3. Узнаем список и названия полей

---

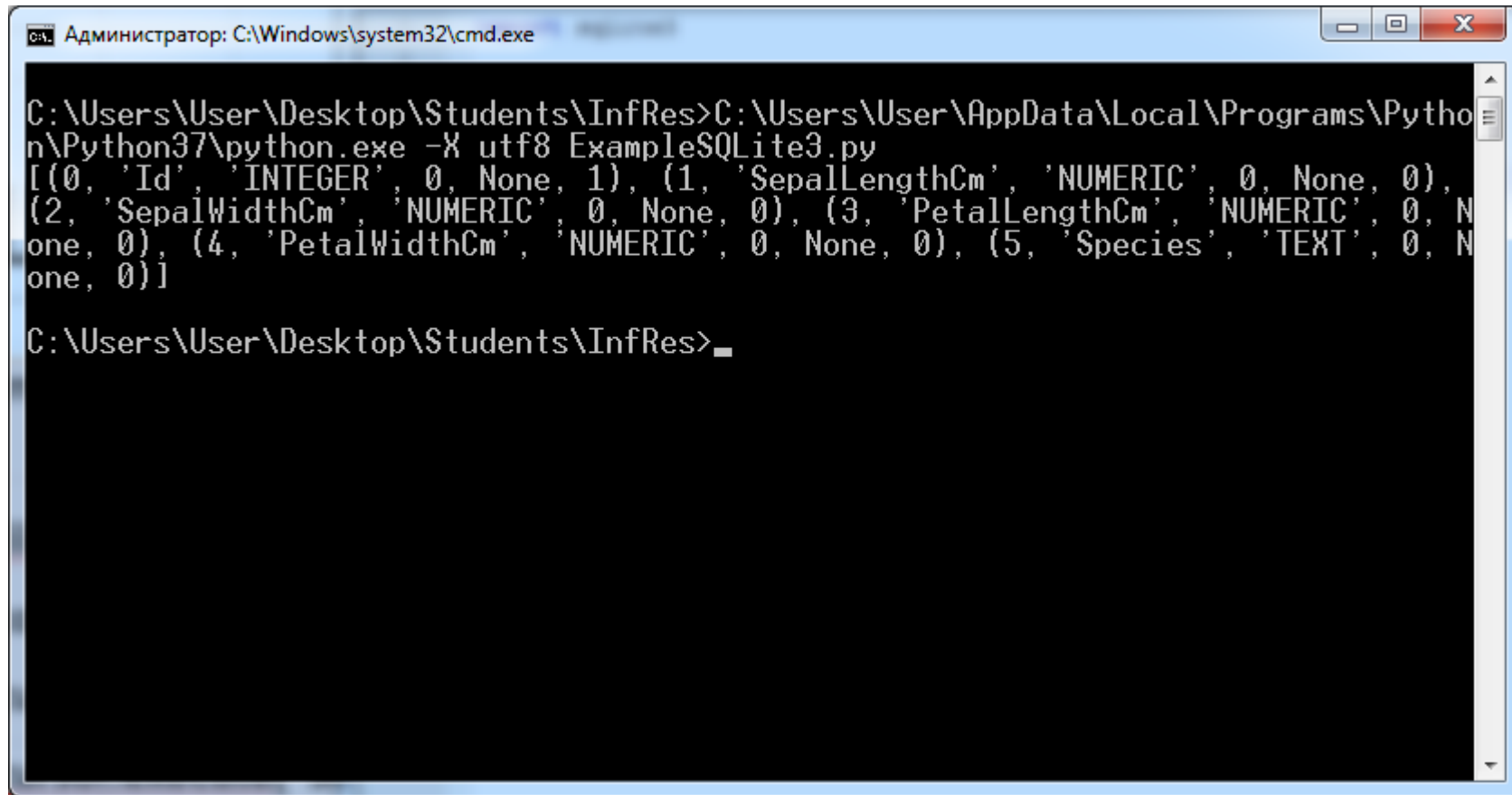
```
import sqlite3

file_path = 'database.sqlite'
conn = sqlite3.connect(file_path)
cursor = conn.cursor()
cursor.execute('PRAGMA table_info(Iris)')
data = cursor.fetchall()
print(data)
conn.close()
```



# Результат

---



A screenshot of a Windows command prompt window titled "Администратор: C:\Windows\system32\cmd.exe". The prompt shows the execution of a Python script: `C:\Users\User\Desktop\Students\InfRes>C:\Users\User\AppData\Local\Programs\Python\Python37\python.exe -X utf8 ExampleSQLite3.py`. The output of the script is a list of tuples representing database schema information: `[(0, 'Id', 'INTEGER', 0, None, 1), (1, 'SepalLengthCm', 'NUMERIC', 0, None, 0), (2, 'SepalWidthCm', 'NUMERIC', 0, None, 0), (3, 'PetalLengthCm', 'NUMERIC', 0, None, 0), (4, 'PetalWidthCm', 'NUMERIC', 0, None, 0), (5, 'Species', 'TEXT', 0, None, 0)]`. The prompt then returns to the command line: `C:\Users\User\Desktop\Students\InfRes>_`.

```
Администратор: C:\Windows\system32\cmd.exe

C:\Users\User\Desktop\Students\InfRes>C:\Users\User\AppData\Local\Programs\Python\Python37\python.exe -X utf8 ExampleSQLite3.py
[(0, 'Id', 'INTEGER', 0, None, 1), (1, 'SepalLengthCm', 'NUMERIC', 0, None, 0),
(2, 'SepalWidthCm', 'NUMERIC', 0, None, 0), (3, 'PetalLengthCm', 'NUMERIC', 0, None, 0), (4, 'PetalWidthCm', 'NUMERIC', 0, None, 0), (5, 'Species', 'TEXT', 0, None, 0)]

C:\Users\User\Desktop\Students\InfRes>_
```

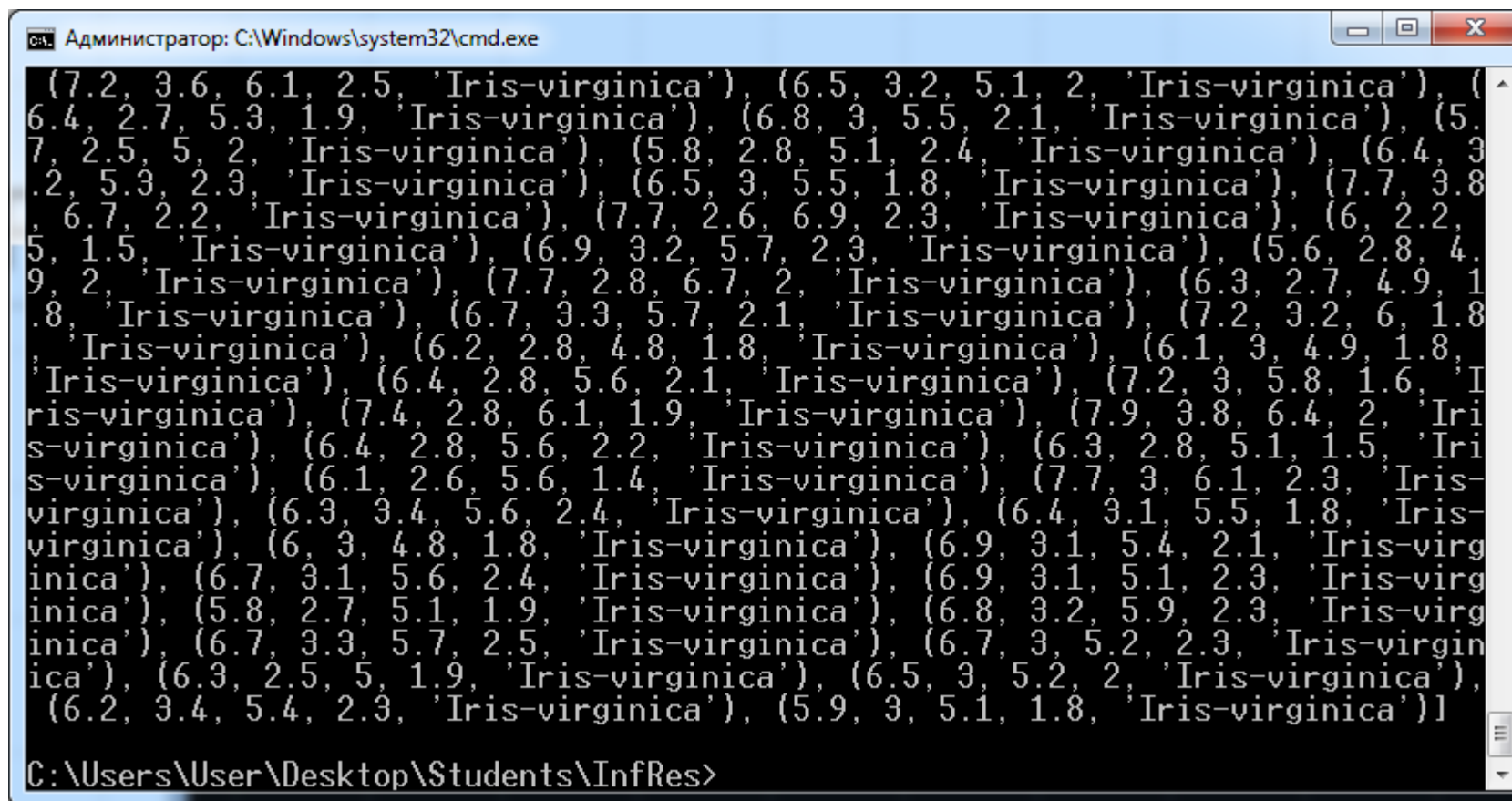
# Шаг 4. Загрузим данные

---

```
import sqlite3
import json

file_path = 'database.sqlite'
conn = sqlite3.connect(file_path)
cursor = conn.cursor()
cursor.execute(
'SELECT SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species FROM Iris ORDER BY id')
data = cursor.fetchall()
print(data)
conn.close()
```

# Результат



```
Администратор: C:\Windows\system32\cmd.exe
(7.2, 3.6, 6.1, 2.5, 'Iris-virginica'), (6.5, 3.2, 5.1, 2, 'Iris-virginica'), (
6.4, 2.7, 5.3, 1.9, 'Iris-virginica'), (6.8, 3, 5.5, 2.1, 'Iris-virginica'), (5.
7, 2.5, 5, 2, 'Iris-virginica'), (5.8, 2.8, 5.1, 2.4, 'Iris-virginica'), (6.4, 3
.2, 5.3, 2.3, 'Iris-virginica'), (6.5, 3, 5.5, 1.8, 'Iris-virginica'), (7.7, 3.8
, 6.7, 2.2, 'Iris-virginica'), (7.7, 2.6, 6.9, 2.3, 'Iris-virginica'), (6, 2.2,
5, 1.5, 'Iris-virginica'), (6.9, 3.2, 5.7, 2.3, 'Iris-virginica'), (5.6, 2.8, 4.
9, 2, 'Iris-virginica'), (7.7, 2.8, 6.7, 2, 'Iris-virginica'), (6.3, 2.7, 4.9, 1
.8, 'Iris-virginica'), (6.7, 3.3, 5.7, 2.1, 'Iris-virginica'), (7.2, 3.2, 6, 1.8
, 'Iris-virginica'), (6.2, 2.8, 4.8, 1.8, 'Iris-virginica'), (6.1, 3, 4.9, 1.8,
'Iris-virginica'), (6.4, 2.8, 5.6, 2.1, 'Iris-virginica'), (7.2, 3, 5.8, 1.6, 'I
ris-virginica'), (7.4, 2.8, 6.1, 1.9, 'Iris-virginica'), (7.9, 3.8, 6.4, 2, 'Iri
s-virginica'), (6.4, 2.8, 5.6, 2.2, 'Iris-virginica'), (6.3, 2.8, 5.1, 1.5, 'Iri
s-virginica'), (6.1, 2.6, 5.6, 1.4, 'Iris-virginica'), (7.7, 3, 6.1, 2.3, 'Iris-
virginica'), (6.3, 3.4, 5.6, 2.4, 'Iris-virginica'), (6.4, 3.1, 5.5, 1.8, 'Iris-
virginica'), (6, 3, 4.8, 1.8, 'Iris-virginica'), (6.9, 3.1, 5.4, 2.1, 'Iris-virg
inica'), (6.7, 3.1, 5.6, 2.4, 'Iris-virginica'), (6.9, 3.1, 5.1, 2.3, 'Iris-virg
inica'), (5.8, 2.7, 5.1, 1.9, 'Iris-virginica'), (6.8, 3.2, 5.9, 2.3, 'Iris-virg
inica'), (6.7, 3.3, 5.7, 2.5, 'Iris-virginica'), (6.7, 3, 5.2, 2.3, 'Iris-virgin
ica'), (6.3, 2.5, 5, 1.9, 'Iris-virginica'), (6.5, 3, 5.2, 2, 'Iris-virginica'),
(6.2, 3.4, 5.4, 2.3, 'Iris-virginica'), (5.9, 3, 5.1, 1.8, 'Iris-virginica'))

C:\Users\User\Desktop\Students\InfRes>
```

# Шаг 5. Немного преобразуем данные

---

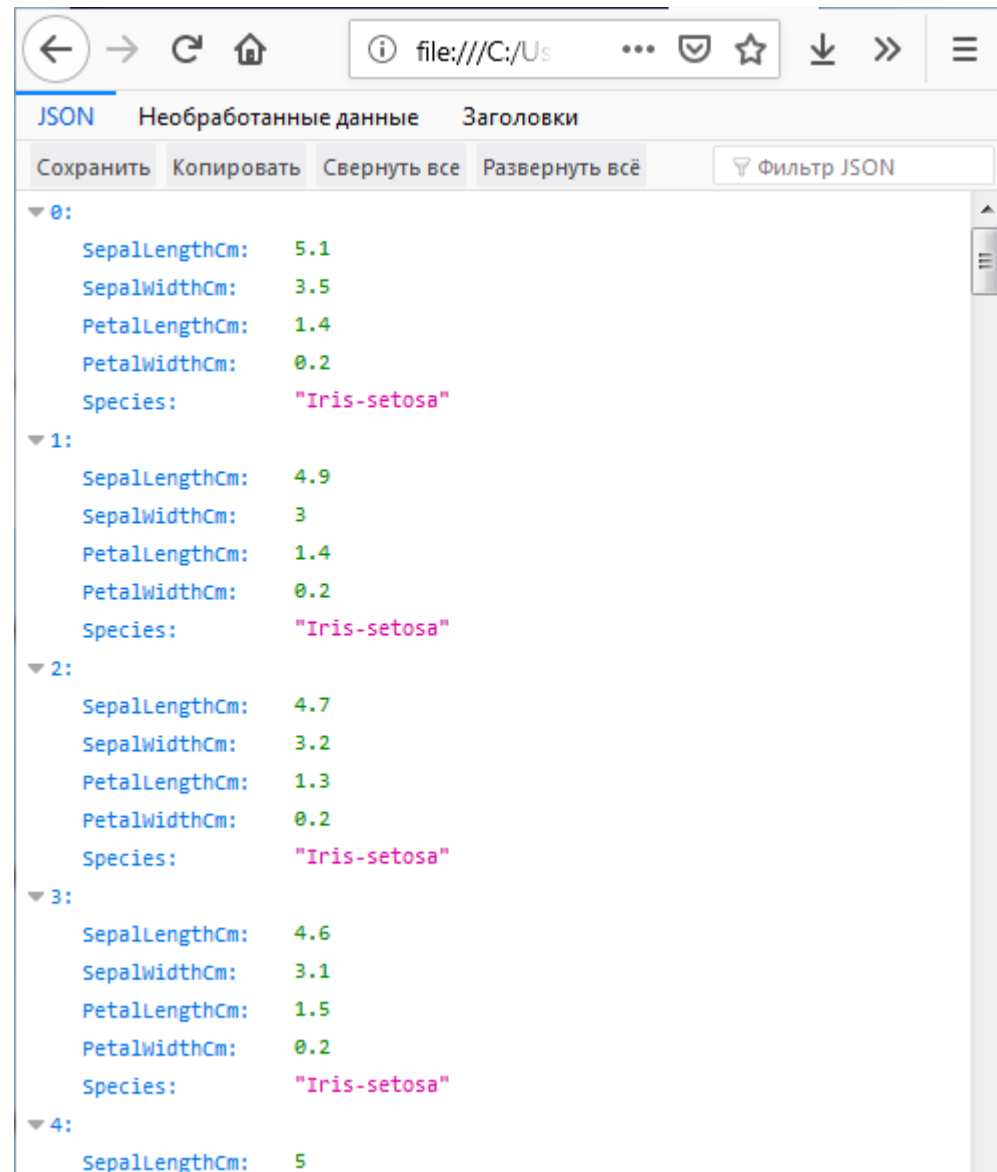
```
# Преобразуем list в dictionary
# Можно было использовать row_factory, но для лучшего понимания проще так
marked_data = []
for row in data:
    mrow = {"SepalLengthCm": row[0], "SepalWidthCm": row[1],
            "PetalLengthCm": row[2], "PetalWidthCm": row[3], "Species": row[4]}
    marked_data.append(mrow)
```

# Шаг 6. Запись в JSON

---

```
with open("data_file.json", "w") as write_file:  
    json.dump(marked_data, write_file)
```

# Итоговый результат



# Работа с PostgreSQL в Python

---

Для работы в языке Python с СУБД PostgreSQL используется библиотека **psycopg2**. Для установки данной библиотеки можно использовать модуль `pip`:

```
python.exe -m pip install psycopg2
```

Принципы работы с данной библиотекой похожи на принципы работы с SQLite.

В качестве примера напишем программу, печатающую содержимое таблицы `regions`.

# Пример

---

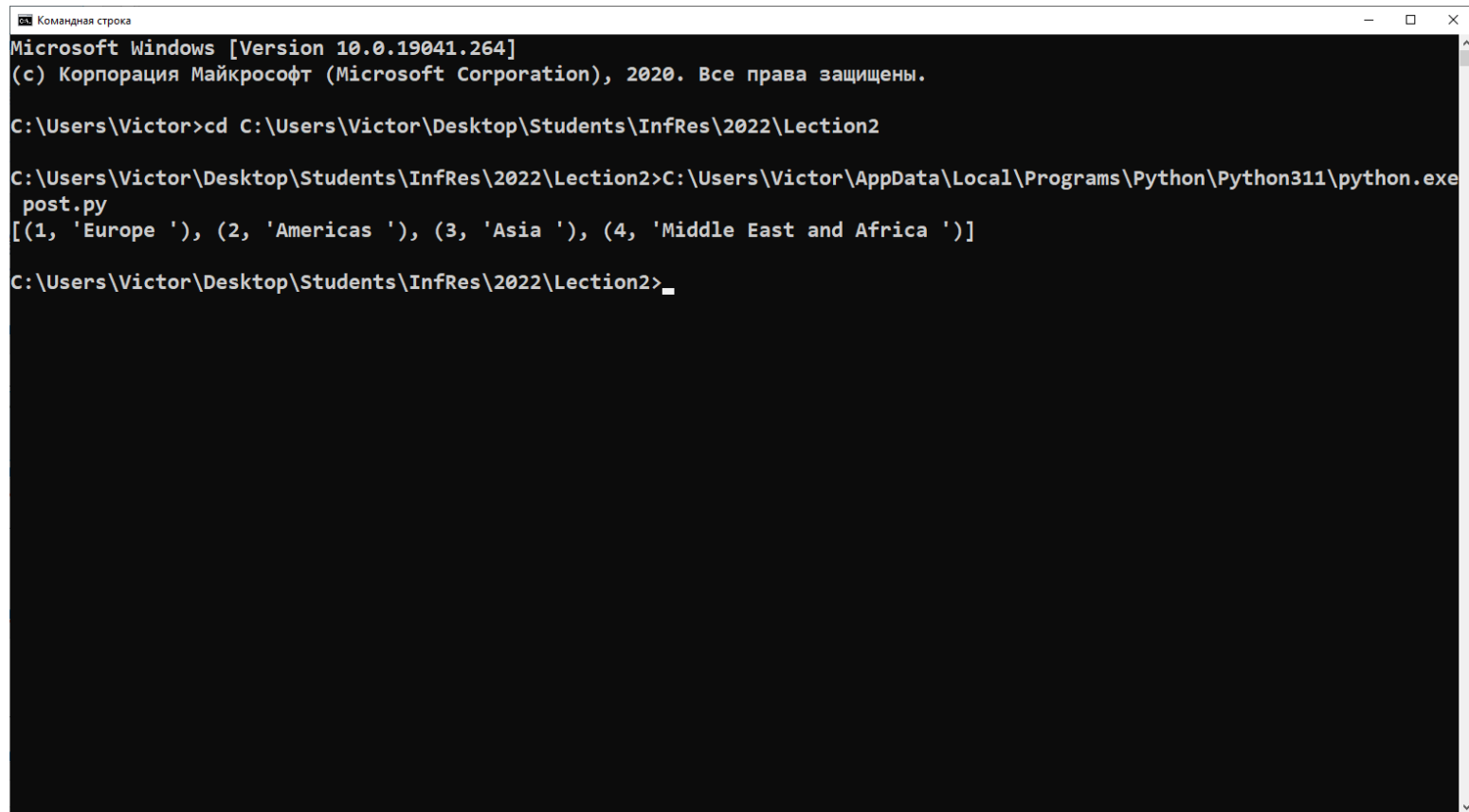
```
import psycopg2

conn = psycopg2.connect(dbname = 'postgres', user = 'postgres',
                        password = 'qwerty', host = 'localhost')
cursor = conn.cursor()
cursor.execute('SELECT * FROM regions')
records = cursor.fetchall()
print(records)
cursor.close()
conn.close()
```



# Результат

---



```
Командная строка
Microsoft Windows [Version 10.0.19041.264]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Users\Victor>cd C:\Users\Victor\Desktop\Students\InfRes\2022\Lec2\ion2

C:\Users\Victor\Desktop\Students\InfRes\2022\Lec2\ion2>C:\Users\Victor\AppData\Local\Programs\Python\Python311\python.exe
post.py
[(1, 'Europe '), (2, 'Americas '), (3, 'Asia '), (4, 'Middle East and Africa ')]

C:\Users\Victor\Desktop\Students\InfRes\2022\Lec2\ion2>
```

# Часть 4

---

API

# API

---

Application programming interface (API) или, если говорить по-русски, то программный интерфейс приложения – это некоторый протокол (набор правил) с помощью которого приложение или его компоненты взаимодействуют с другими приложениями или компонентами.

В нашем курсе мы будем рассматривать только API для обмена данными между информационными системами. Поэтому такие API, как, например, Windows API или Linux Kernel API нам не интересны.

Грубо говоря, API в нашем случае – это описание того, как другой программе надо обратиться к приложению, чтобы получить от него в автоматическом (без работы оператора) режиме информацию и/или выполнить изменения какой-либо информации.

Естественно, протоколу должна соответствовать работающая программная часть.

В общем случае подобных протоколов могут придумать тысячи. Поэтому для разработки API в последнее время вводятся различные стандарты. Наиболее известны из них SOAP и REST.

# SOAP API

---

SOAP расшифровывается как *Simple Object Access Protocol* — простой протокол доступа к объектам. Проще говоря SOAP – это протокол обмена структурированными сообщениями в распределённой вычислительной среде (источник определения – википедия).

SOAP является развитием XML-RPC протокола. Обмен сообщениями в данном протоколе построен на основе передачи XML-документов с определённой структурой и основными тегами.

Подробно о стандарте SOAP можно прочесть на сайте консорциума W3C [6, 7].

# Элементы SOAP

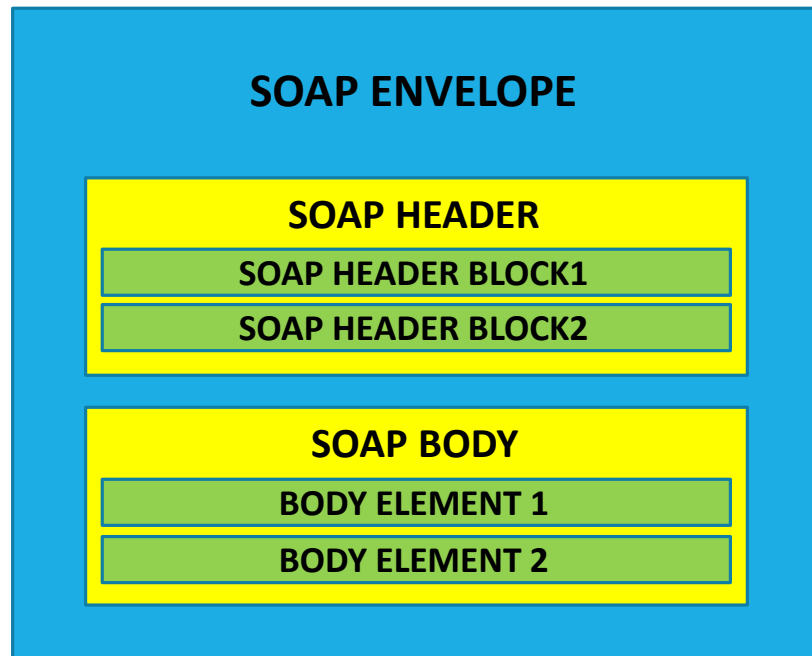
---

Стандартное SOAP-сообщение (SOAP message) – это XML-документ, передаваемый в рамках SOAP-протокола, который состоит из следующих компонент:

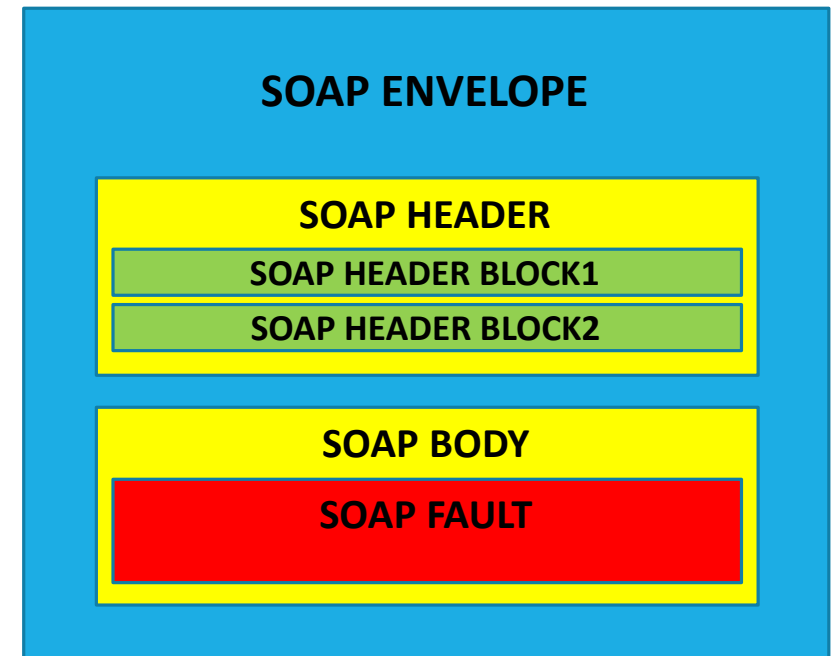
- Обертки (**Envelope** )
- Заголовка (**Header**), который может включать в себя Блоки заголовка (**Header Blocks**).
- Тела (**Body**)
- Ошибки (**Fault**), расположенные внутри Body.

# Структура SOAP Message

---



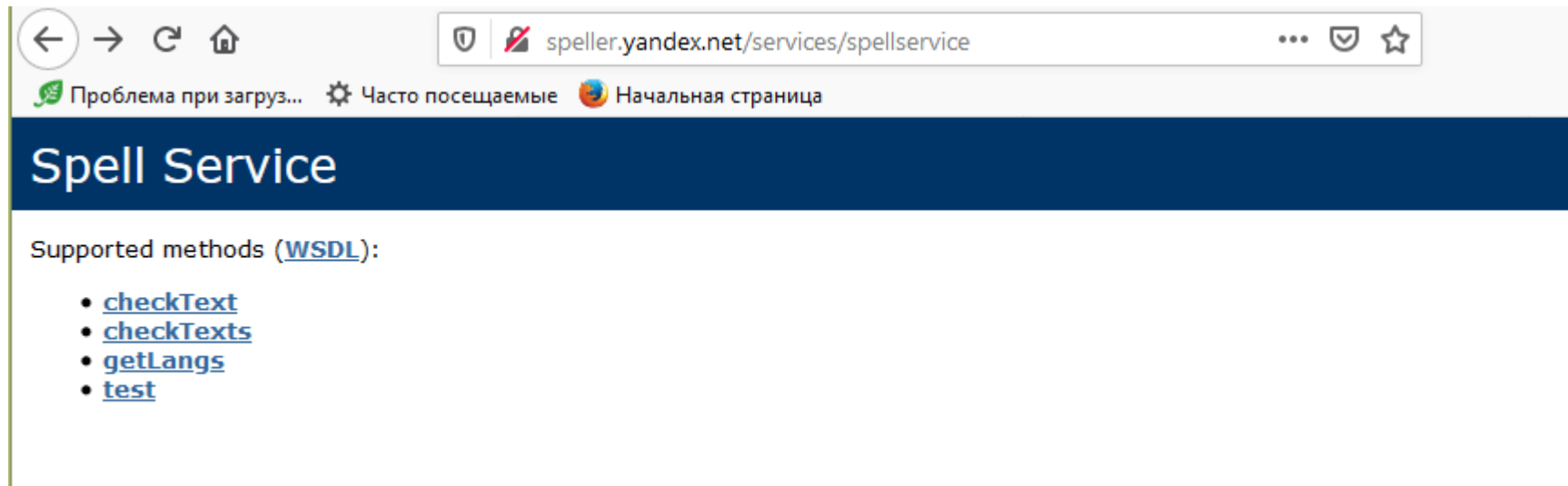
Успешное обращение



Ошибочное обращение

# Пример. Yandex Spell Service

В качестве примера SOAP API рассмотрим сервис проверки правописания от Yandex. Он доступен по адресу <http://speller.yandex.net/services/spellservice>. Данный сервис можно использовать по протоколу SOAP.



# Как узнать какие есть методы у API?

---

Когда кто-то создаёт новое API он должен также обеспечить какой-то механизм описания того, какие действия можно делать с данным API и какие атрибуты у его методов. Для решения данной задачи был введён стандарт WSDL.

Чтобы лучше понять данный момент рассмотрим следующую аналогию. Вы договорились со своим компаньоном, что будете передавать друг другу документы по почте России. Это аналог выбора HTTP-протокола для отправки сообщений.

Затем Вы договорились, что каждое письмо будет в бумажном конверте. Причём внутри конверта будет два вложения: описание пересылаемого документа и сам документ. Это аналог выбора SOAP для базовой формы передаваемых сообщений.

Но теперь Вам надо определить, какие документы Вы будете посылать другу и какие документы он должен будет прислать в ответ. Это и есть WSDL.



# WSDL

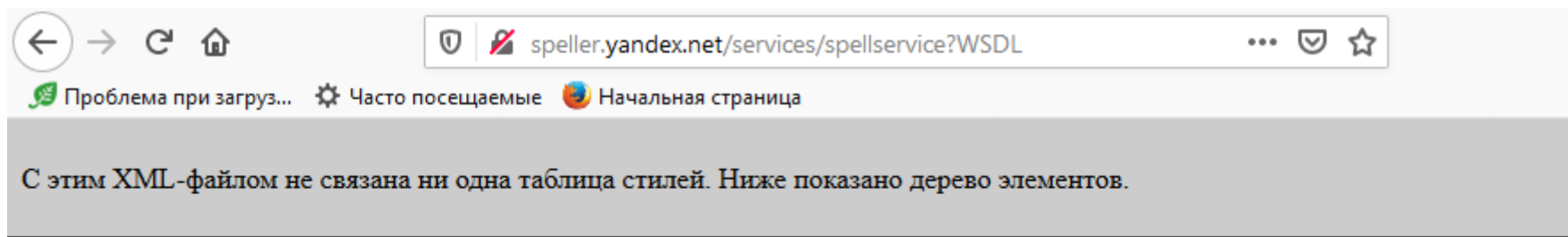
---

WSDL — *Web Services Description Language* — язык описания веб-сервисов и доступа к ним, основанный на языке XML (источник — Википедия).

WSDL — это некоторый стандарт, позволяющий при помощи всё того же языка XML и жёстко специфицированного набора его тегов описать доступные сервисы API. Подробно о WSDL можно прочесть на сайте консорциума W3C [8]. Стоит отметить, что наиболее последняя версия WSDL — 2.0 не является наиболее популярной. Многие сервисы используют версию WSDL 1.2 (см. [9]).

Для простоты рассмотрим WSDL 1.2.

# WSDL-описание API Yandex Spell Service



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://speller.yandex.net/services/spellservice">
  <wsdl:types/>
  <wsdl:message name="checkTextSoapIn"/>
  <wsdl:message name="checkTextSoapOut"/>
  <wsdl:message name="checkTextsSoapIn"/>
  <wsdl:message name="checkTextsSoapOut"/>
  <wsdl:portType name="SpellServiceSoap"/>
  <wsdl:binding name="SpellServiceSoap" type="tns:SpellServiceSoap"/>
  <wsdl:binding name="SpellServiceSoap12" type="tns:SpellServiceSoap"/>
  <wsdl:service name="SpellService"/>
</wsdl:definitions>
```

# Элементы WSDL 1.2

---

- Описание типов – type description component – `wsdl:types`
- Описание сообщений – message description component – `wsdl:message`
- Описание операций – portType description component – `wsdl:portType`
- Набор операций – serviceType description component – `wsdl:serviceType`
- Описание способов доставки сообщений – binding description component – `wsdl:binding`
- Описание сервиса – service description component – `wsdl:service`

Рассмотрим их чуть подробнее на примере с API Yandex Spell Service.

# wsdl:types

---

Элемент `wsdl:types` позволяет с помощью какого-либо стандарта описать типы данных, передаваемых сообщений. Обычно используется стандарт **XML Schema definition (XSD)**.

Чтобы было понятно, рассмотрим следующую задачу нам надо отправить запрос к сервису и получить от него ответ. При этом ответ может быть положительным, а могут быть и ошибки двух видов. Итого мы получаем 4 вида сообщений: запрос, удачный ответ, ответ ошибка первого вида и ответ ошибка второго вида. Для каждого из этих сообщений должно быть строго описано, что в них может входить и как это должно быть оформлено. Именно такие вещи и описывают в элементе `wsdl:types`.

# wsdl:types API Yandex Spell Service

---

```
<wsdl:definitions targetNamespace="http://speller.yandex.net/services/spellservice">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://speller.yandex.net/services/spellservice">
      <s:element name="CheckTextRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="text" type="s:string"/>
          </s:sequence>
          <s:attribute name="lang" type="s:string"/>
          <s:attribute name="options" type="s:int" use="optional" default="0"/>
          <s:attribute name="format" type="s:string" use="optional" default=""/>
        </s:complexType>
      </s:element>
      <s:element name="CheckTextResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="SpellResult" type="tns:SpellResult"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
</wsdl:definitions>
```

# wsdl:message

---

Типы данных могут повторяться. Поэтому, чтобы не дублировать их, есть возможность одни и те же типы сообщений связать с разными реальными обменов сообщениями. Элемент `wsdl:message` позволяет сопоставить тип реальному обмену и, фактически, заявить, что данный обмен будет.

# wsdl:message API Yandex Spell Service

---

```
-<wsdl:definitions targetNamespace="http://speller.yandex.net/services/spellservice">
  +<wsdl:types></wsdl:types>
  -<wsdl:message name="checkTextSoapIn">
    <wsdl:part name="parameters" element="tns:CheckTextRequest"/>
  </wsdl:message>
  -<wsdl:message name="checkTextSoapOut">
    <wsdl:part name="parameters" element="tns:CheckTextResponse"/>
  </wsdl:message>
  -<wsdl:message name="checkTextsSoapIn">
    <wsdl:part name="parameters" element="tns:CheckTextsRequest"/>
  </wsdl:message>
  -<wsdl:message name="checkTextsSoapOut">
    <wsdl:part name="parameters" element="tns:CheckTextsResponse"/>
  </wsdl:message>
  +<wsdl:portType name="SpellServiceSoap"></wsdl:portType>
  +<wsdl:binding name="SpellServiceSoap" type="tns:SpellServiceSoap"></wsdl:binding>
  +<wsdl:binding name="SpellServiceSoap12" type="tns:SpellServiceSoap"></wsdl:binding>
  +<wsdl:service name="SpellService"></wsdl:service>
</wsdl:definitions>
```

# Типы взаимодействия в WSDL

---

Различают следующие типы взаимодействия в WSDL:

- Однонаправленный запрос (one-way) — конечная точка принимает сообщение.
- Запрос-ответ (request-response) — конечная точка принимает сообщение и отправляет связанное сообщение.
- Просьба-ответ (solicit-response) — конечная точка отправляет сообщение и получает связанное сообщение.
- Уведомление (notification) — конечная точка отправляет сообщение.

Так как типы взаимодействия могут быть разными, то надо как-то уметь задавать описание данного процесса. Для этого существует элемент описания операции `wsdl:portType`.



# wsdl:portType

---

`wsdl:portType` – это элемент, содержащий описание типа операции и связанных с ним сообщений.

Внутри `wsdl:portType` состоит из набора тегов `wsdl:input`, `wsdl:output` и `wsdl:fault`, обёрнутых в тег `wsdl:operation`. В зависимости от их порядка получаем разный тип операции:

- Однонаправленный запрос: `wsdl:input`.
- Запрос-ответ: `wsdl:input`, `wsdl:output`, [`wsdl:fault`].
- Просьба-ответ: `wsdl:output`, `wsdl:input`, [`wsdl:fault`].
- Уведомление: `wsdl:output`.

# wsdl:portType API Yandex Spell Service

---

```
-<wsdl:definitions targetNamespace="http://speller.yandex.net/services/spellservice">
  +<wsdl:types></wsdl:types>
  +<wsdl:message name="checkTextSoapIn"></wsdl:message>
  +<wsdl:message name="checkTextSoapOut"></wsdl:message>
  +<wsdl:message name="checkTextsSoapIn"></wsdl:message>
  +<wsdl:message name="checkTextsSoapOut"></wsdl:message>
  -<wsdl:portType name="SpellServiceSoap">
    -<wsdl:operation name="checkText">
      <wsdl:input message="tns:checkTextSoapIn"/>
      <wsdl:output message="tns:checkTextSoapOut"/>
    </wsdl:operation>
    -<wsdl:operation name="checkTexts">
      <wsdl:input message="tns:checkTextsSoapIn"/>
      <wsdl:output message="tns:checkTextsSoapOut"/>
    </wsdl:operation>
  </wsdl:portType>
  +<wsdl:binding name="SpellServiceSoap" type="tns:SpellServiceSoap"></wsdl:binding>
  +<wsdl:binding name="SpellServiceSoap12" type="tns:SpellServiceSoap"></wsdl:binding>
  +<wsdl:service name="SpellService"></wsdl:service>
</wsdl:definitions>
```

# wsdl:binding

---

wsdl:binding – это, грубо говоря, характеристики транспортного уровня для описания операций. Фактически, данные теги содержат часть описания процесса работы операций, не вошедшую в тег wsdl:portType.

# wsdl:binding API Yandex Spell Service

---

```

+<wsdl:portType name="SpellServiceSoap" />
-<wsdl:binding name="SpellServiceSoap" type="tns:SpellServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  -<wsdl:operation name="checkText">
    <soap:operation soapAction="http://speller.yandex.net/services/spellservice/checkText" style="document"/>
    -<wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    -<wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  -<wsdl:operation name="checkTexts">
    <soap:operation soapAction="http://speller.yandex.net/services/spellservice/checkTexts" style="document"/>
    -<wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    -<wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
+<wsdl:binding name="SpellServiceSoap12" type="tns:SpellServiceSoap"></wsdl:binding>

```

# wsdl:service

---

wsdl:service – это, фактически, группировка нескольких wsdl:binding в единый пакет операций – сервис.

# wsdl:service API Yandex Spell Service

---

```
-<wsdl:definitions targetNamespace="http://speller.yandex.net/services/spellservice">
  +<wsdl:types></wsdl:types>
  +<wsdl:message name="checkTextSoapIn"></wsdl:message>
  +<wsdl:message name="checkTextSoapOut"></wsdl:message>
  +<wsdl:message name="checkTextsSoapIn"></wsdl:message>
  +<wsdl:message name="checkTextsSoapOut"></wsdl:message>
  +<wsdl:portType name="SpellServiceSoap"></wsdl:portType>
  +<wsdl:binding name="SpellServiceSoap" type="tns:SpellServiceSoap"></wsdl:binding>
  +<wsdl:binding name="SpellServiceSoap12" type="tns:SpellServiceSoap"></wsdl:binding>
  -<wsdl:service name="SpellService">
    -<wsdl:port name="SpellServiceSoap" binding="tns:SpellServiceSoap">
      <soap:address location="http://speller.yandex.net/services/spellservice"/>
    </wsdl:port>
    -<wsdl:port name="SpellServiceSoap12" binding="tns:SpellServiceSoap12">
      <soap12:address location="http://speller.yandex.net/services/spellservice"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

# Работа с SOAP API на уровне клиента

---

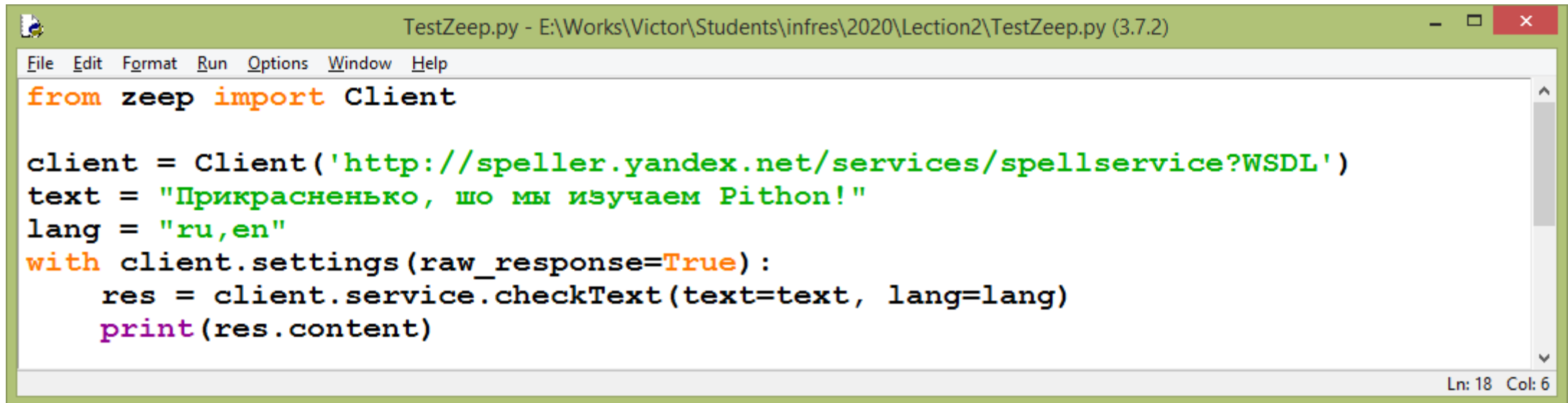
Для обращения к SOAP API в языке Python можно использовать библиотеку zeep [10]. Для её установки используйте:

```
python.exe -m pip install zeep
```

или в Anaconda:

```
conda install zeep
```

# Пример. Проверка правописания 1



```
TestZeep.py - E:\Works\Victor\Students\infres\2020\Lecion2\TestZeep.py (3.7.2)
File Edit Format Run Options Window Help
from zeep import Client

client = Client('http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасненько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    print(res.content)
```

Ln: 18 Col: 6



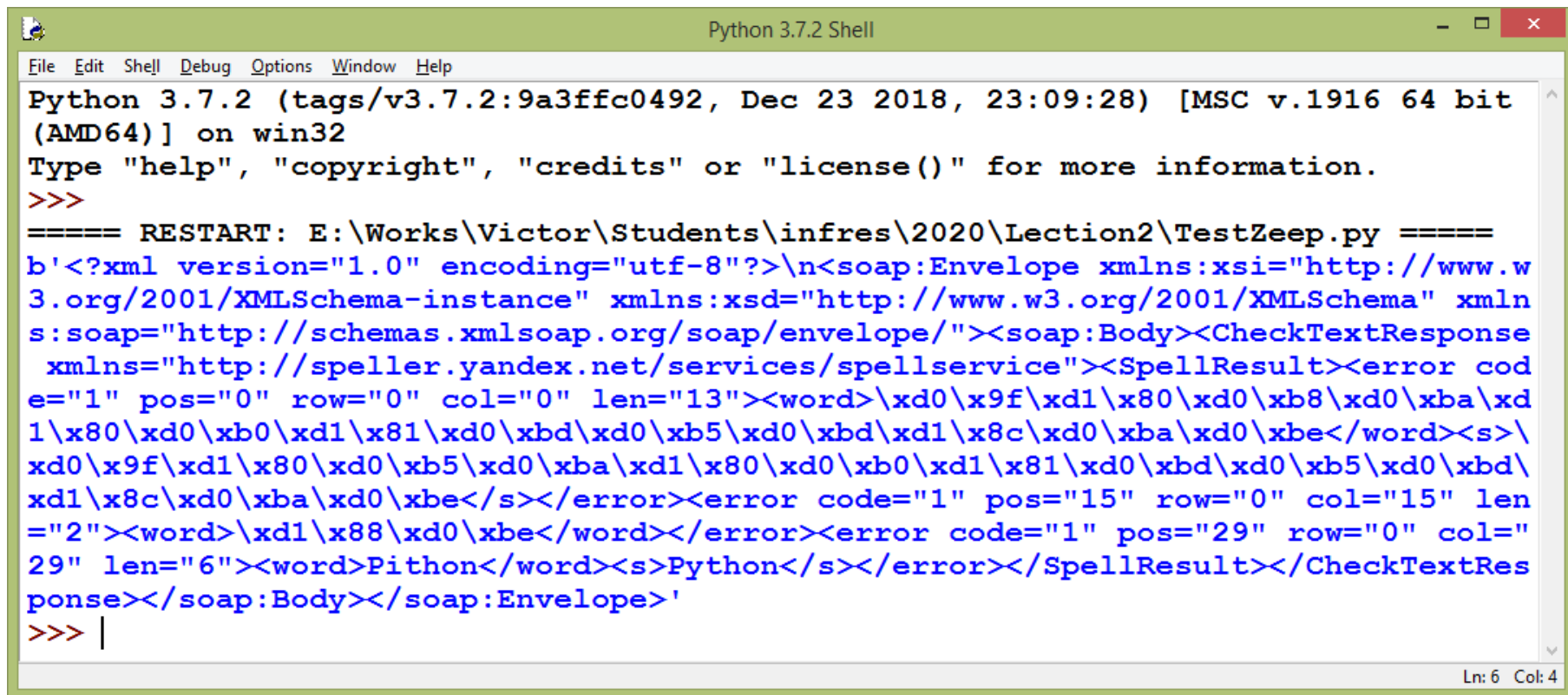
# Пример. Проверка правописания 1

---

```
from zeep import Client

client = Client('http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасненько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    print(res.content)
```

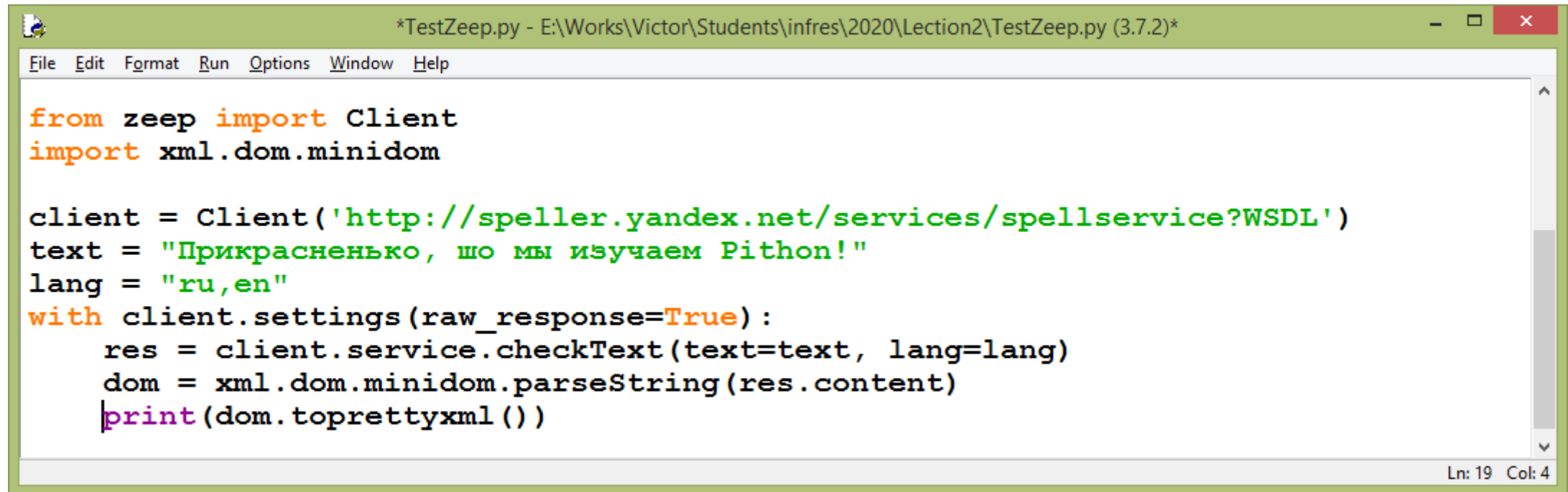
# Результат



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\Works\Victor\Students\infres\2020\Lecion2\TestZeep.py =====
b'<?xml version="1.0" encoding="utf-8"?>\n<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><CheckTextResponse xmlns="http://speller.yandex.net/services/spellservice"><SpellResult><error code="1" pos="0" row="0" col="0" len="13"><word>\xd0\x9f\xd1\x80\xd0\xb8\xd0\xba\xd1\x80\xd0\xb0\xd1\x81\xd0\xbd\xd0\xb5\xd0\xbd\xd1\x8c\xd0\xba\xd0\xbe</word><s>\xd0\x9f\xd1\x80\xd0\xb5\xd0\xba\xd1\x80\xd0\xb0\xd1\x81\xd0\xbd\xd0\xb5\xd0\xbd\xd1\x8c\xd0\xba\xd0\xbe</s></error><error code="1" pos="15" row="0" col="15" len="2"><word>\xd1\x88\xd0\xbe</word></error><error code="1" pos="29" row="0" col="29" len="6"><word>Pithon</word><s>Python</s></error></SpellResult></CheckTextResponse</soap:Body></soap:Envelope>'
>>> |
```

Не очень-то красиво и понятно. Попробуем исправить....

# Пример. Проверка правописания 2



```
*TestZeep.py - E:\Works\Victor\Students\infres\2020\Lecton2\TestZeep.py (3.7.2)*
File Edit Format Run Options Window Help

from zeep import Client
import xml.dom.minidom

client = Client('http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасненько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    dom = xml.dom.minidom.parseString(res.content)
    print(dom.toprettyxml())
```

Ln: 19 Col: 4

# Пример. Проверка правописания 2

---

```
from zeep import Client
import xml.dom.minidom

client = Client('http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасненько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    dom = xml.dom.minidom.parseString(res.content)
    print(dom.toprettyxml())
```

# Результат

---

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <CheckTextResponse xmlns="http://speller.yandex.net/services/spellservice">
      <SpellResult>
        <error code="1" col="0" len="13" pos="0" row="0">
          <word>Прикрасенько</word>
          <s>Прекрасенько</s>
        </error>
        <error code="1" col="15" len="2" pos="15" row="0">
          <word>мо</word>
        </error>
        <error code="1" col="29" len="6" pos="29" row="0">
          <word>Python</word>
          <s>Python</s>
        </error>
      </SpellResult>
    </CheckTextResponse>
  </soap:Body>
</soap:Envelope>
```

# Исправим предложенные коррекции

---

Yandex Spell Service для некоторых слов предложил нам варианты исправления. Давайте напишем программу, которая будет исправлять текст, согласно рекомендации Yandex.

Все найденные ошибки сложены в тегах `error`. Если есть исправление, то внутри тега `error` появляется тег `s`.

Чтобы не искать, в каком месте текста нужно внести изменение используем параметры `pos` и `len` тега `error` (позиция и длина слова).

# Пример. Коррекция

```
*TestZee.py - C:\Users\User\Desktop\Students\InfRes\2020\Lecion2\TestZee.py (3.7.2)*
File Edit Format Run Options Window Help
import xml.dom.minidom

client = Client('http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасенько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    dom = xml.dom.minidom.parseString(res.content)
    print(dom.toprettyxml())
    elems = dom.getElementsByTagName("error")
    for elem in elems:
        pos = elem.getAttribute("pos")
        wlen = elem.getAttribute("len")
        s = elem.getElementsByTagName("s")
        if s:
            txt = s[0]
            txt = txt.firstChild.nodeValue
            text = text[0:int(pos)] + str(txt) + text[int(pos) + int(wlen):-1]
print(text)
```

Ln: 8 Col: 6

# Пример. Коррекция

---

```
from zeep import Client
import xml.dom.minidom
client = Client(
    'http://speller.yandex.net/services/spellservice?WSDL')
text = "Прикрасненько, шо мы изучаем Python!"
lang = "ru,en"
with client.settings(raw_response=True):
    res = client.service.checkText(text=text, lang=lang)
    dom = xml.dom.minidom.parseString(res.content)
    print(dom.toprettyxml())
    elems = dom.getElementsByTagName("error")
```

```
for elem in elems:
    pos = elem.getAttribute("pos")
    wlen = elem.getAttribute("len")
    s = elem.getElementsByTagName("s")
    if s:
        txt = s[0]
        txt = txt.firstChild.nodeValue
        text = text[0:int(pos)] + str(txt) +
            text[int(pos) + int(wlen):-1]
print(text)
```



# Результат

---

```
Прекрасненько, що ми изучаєм Python
```

```
>>>
```

Ln: 1765 Col: 13

# REST API

---

REST (Representational State Transfer) – это некоторый стандарт описания архитектуры прикладных интерфейсов. Мы не будем рассматривать его слишком глубоко, а рассмотрим основы построения REST API.

# REST API на основе HTTP-протокола

---

Для знакомства с понятием REST API мы рассмотрим построение REST API на основе HTTP-протокола передачи данных.

Мы будем считать, что есть четыре основных операции, предоставляемых API: получение информации об объекте, создание нового объекта, изменение объекта, удаление объекта.

Каждой из этих операции соответствует свой тип запроса:

- **GET** – получение
- **POST** – создание
- **PUT** – обновление, модификация
- **DELETE** – удаление

# Полезные ссылки

---

1. <http://www.json.org/> – Сайт формата JSON.
2. <https://www.microsoft.com/ru-ru/download/details.aspx?id=39379> – PowerQuery для Microsoft Excel.
3. <https://www.python.org/downloads/release/python-372/> – здесь можно скачать Python.
4. <https://www.sqlite.org/download.html> – здесь можно скачать SQLite.
5. <https://www.kaggle.com/uciml/iris#database.sqlite> – набор данных со спецификацией ирисов.
6. <https://www.w3.org/TR/soap/>
7. [https://www.w3schools.com/xml/xml\\_soap.asp](https://www.w3schools.com/xml/xml_soap.asp)

# Полезные ссылки

---

8. <https://www.w3.org/TR/wsdl/>
9. <https://www.w3.org/TR/2002/WD-wsdl12-20020709/#language-syntax>
10. <https://python-zee.readthedocs.io/en/master/>
11. <https://sitkodenis.ru/useful-api-services/>
12. <http://www.la2q.com/2017/03/27/helpful-api-services/>
13. <https://github.com/public-apis/public-apis>
14. <https://www.programmableweb.com/apis/directory>