

# Работа с разными форматами данных

Елена Никитина

Руководитель проектов ООО «Аналитические программные решения»

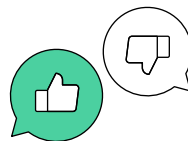


# Проверка связи





## Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти



## Поставьте в чат:

-  если меня видно и слышно
-  если нет

# Рекомендации

## → Если смотрите с компьютера

- Используйте браузеры **Google Chrome** или **Microsoft Edge**
- Если есть проблемы с изображением или звуком, обновите страницу — **F5**

## → Если смотрите с мобильного телефона или планшета

- Перейдите с мобильного интернет-соединения на **Wi-Fi**
- Если есть проблемы с изображением или звуком, перезапустите приложение **МТС Линк** на телефоне
- Предварительно проверьте, подходит ли ваше устройство для подключения к вебинару, по [ссылке](#)

# Елена Никитина

## О спикере:

- руководитель проектов в ИТ-компании, стаж в отрасли — 19 лет
- 5 лет в разработке летающей робототехники и беспилотных систем
- преподаёт и руководит проектной деятельностью студентов и школьников по программированию и робототехнике в МАИ, «Сириусе» и на Национальной технологической олимпиаде



# Сегодня вы узнаете

- 1 Какие распространённые форматы данных существуют
- 2 Как работать с CSV
- 3 Как работать с JSON
- 4 Как работать с YAML
- 5 Как работать с XML
- 6 Что такое кодировки и как решать распространённые проблемы с ними



# План занятия

- 1 Форматы данных
- 2 Формат CSV
- 3 Формат JSON
- 4 Формат YAML
- 5 Формат XML
- 6 Проблема кодировок
- 7 Итоги и ваши вопросы



# Форматы данных

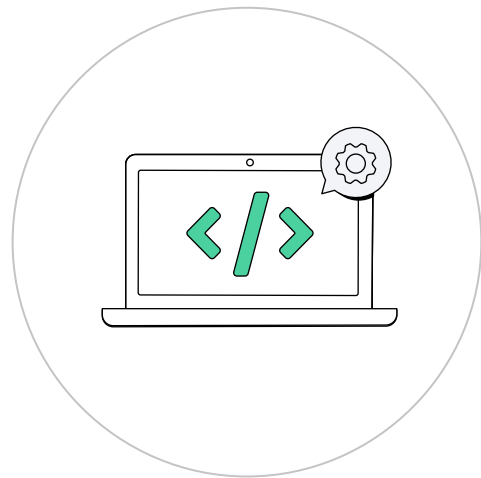
1



# Плоские форматы

Плоские данные представляют информацию в виде одного уровня (таблицы), где каждая запись содержит поля без вложенности.

```
id,name,age  
1,Alice,25  
2,Bob,30
```





# Плоские форматы

→ CSV (Comma-Separated Values)

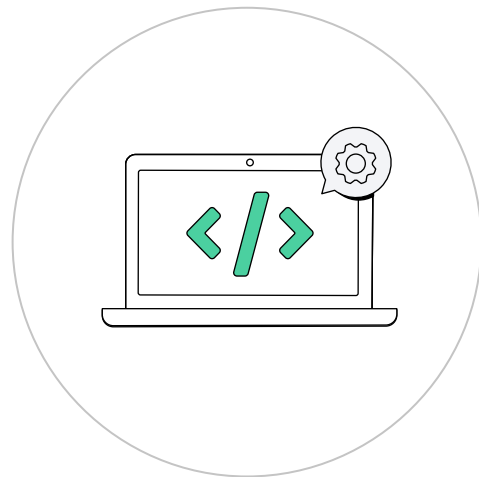
→ Реляционные таблицы (SQL)

## Плюсы:

- Простота чтения и записи
- Подходит для табличных данных

## Минусы:

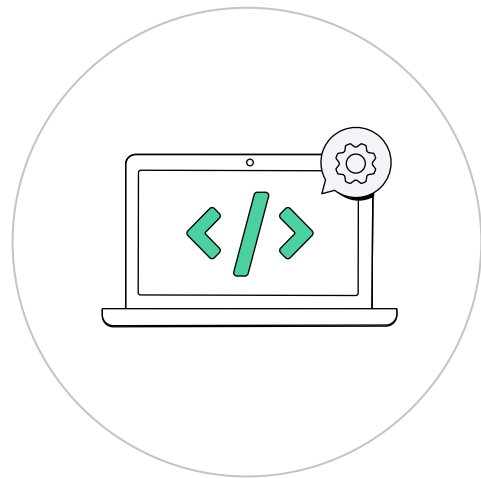
- Не поддерживает сложные структуры (вложенные объекты)
- Избыточность при хранении связанных данных



# Древовидные форматы

Древовидные данные организуют информацию в иерархическую структуру (узлы и подузлы), где одни элементы могут содержать другие.

```
<users>
  <user>
    <id>1</id>
    <name>Alice</name>
    <age>25</age>
  </user>
</users>
```



# Древовидные форматы

→ JSON

→ YAML

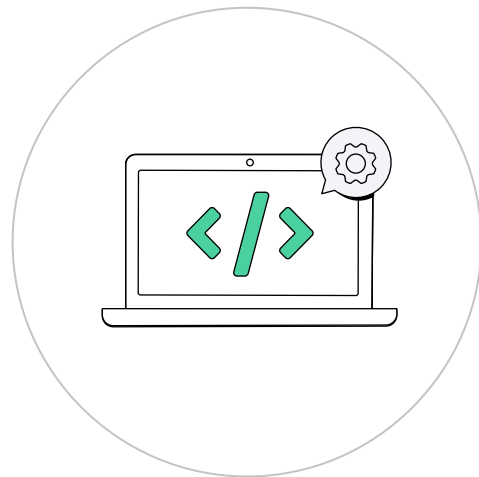
→ XML

## Плюсы:

- Поддержка сложных структур
- Минимизация дублирования
- Гибкость в представлении данных

## Минусы:

- Сложнее обрабатывать (нужны парсеры)
- Большой объём данных из-за мета-разметки (особенно XML)

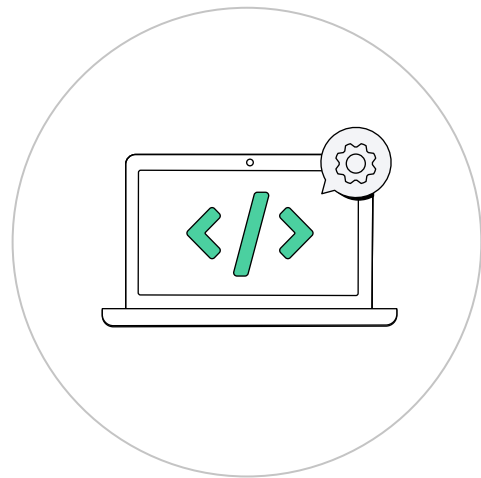


# Сериализация и десериализация

Сериализация — процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

Эта операция предназначена для того, чтобы сохранить состояния объекта для последующего воссоздания при необходимости.

Обратный процесс называется десериализацией.



# Тренировочные данные: новости в RSS

Code Beautify

Форматировщик JSON Форматировщик XML Калькуляторы JSON-Конструктор Последние ссылки Карта сайта Фавориты 

Вход

Средство просмотра RSS

Добавить в Избранное Новое Сохранить и поделиться

Образец

Файл URL-адрес

Автоматическое Обновление

Запуск / Просмотр

RSS в JSON

HTML - код

Скачать

Стр: 82, Кол: 6

размер: 6,86 КБ

Выходной сигнал

Процедурная генерация двухмерной полигональной карты

Tue, 25 Mar 2025 08:58:21 +0300

Здесь я рассмотрю конкретную задачу по генерации, её решение и опишу ключевые использованные принципы. Пишу эту статью для того, чтобы поделиться идеями и опытом, которых мне не хватало, когда я взялся за дело две недели назад. Я не буду делать полный разбор проекта, а лишь опишу и визуализирую принцип.

Read More

pyarsing - 3.2.3

Tue, 25 Mar 2025 08:36:08 +0300

Python модуль для синтаксического анализа. Скачать можно по ссылке: <https://pypi.python.org/pypi/pyarsing/>

Read More

Бэкtesting торговых стратегий на Python с помощью Numba. Когда перевод расчетов на GPU действительно оправдан?

Tue, 25 Mar 2025 08:24:00 +0300

Бэкtesting — ключевой процесс в алгоритмической торговле. Он позволяет проверить стратегию на исторических данных, прежде чем запускать её в реальной торговле. Однако, чем больше данных и сложнее логика стратегии, тем дольше времени занимает вычисления.

Read More

Как мы приручали «Торнадо» для опасного производства

Tue, 25 Mar 2025 06:06:41 +0300

Как отправить работа на автономное патрулирование и научить его «видеть»? Разбираем реальный кейс создания роботизированной системы мониторинга на базе отечественной платформы: выбор оборудования, интеграция ИИ и примеры кода. Делимся опытом, как научить работа собирать данные, обходить препятствия и почему отказались от модной робособаки.

Read More

code

Данные: <http://pythondigest.ru/rss/>

Онлайн просмотр ленты RSS: <https://codebeautify.org/rssviewer>

# Формат CSV

2

# Формат CSV

- 1 title,link,description,pubDate
- 2 Как не править Python тесты,[https://habr.com/ru/post/502278/?utm\\_campaign=502278&utm\\_source=habrahabr&utm\\_medium=rss](https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&utm_medium=rss),"<p>И вынести тестируемые результаты вне кода. Это статья об автоматизации и увеличения удобства тестирования на Python.</p><p>У меня был проект, который разрабатывался уже несколько лет. В проекте отсутствовали тесты. А также у него были активные зависимости от других команд, которые также влияли на результат.</p><p>Регрессионное тестирование было одним из шагов для более уверенной разработки. Его суть в сравнении вычисленных данных с последним канонизированным результатом работы программы.</p><p>Результаты выполнения можно проверять в python коде тестов. Это близко к контексту выполнения и зачастую удобно.</p>","Wed, 20 May 2020 11:25:18 +0300"
- 3 Как построить диаграмму Венна с 50 кругами? Визуализация множеств и история моего Python-проекта с открытым кодом,[https://habr.com/ru/post/501924/?utm\\_campaign=501924&utm\\_source=habrahabr&utm\\_medium=rss](https://habr.com/ru/post/501924/?utm_campaign=501924&utm_source=habrahabr&utm_medium=rss),"<p>Сегодня хочу рассказать вам про задачу

1	title	link	description	pubDate
2	Как не править Python тесты	<a href="https://habr.com/ru/post/502278">https://habr.com/ru/post/502278</a>	<p>И вынести тестируемые резуль	20.05.2020 11:25
3	Как построить диаграмму Венна с 50 кругами? Визуализация мно	<a href="https://habr.com/ru/post/501924">https://habr.com/ru/post/501924</a>	<p>Сегодня хочу рассказать вам п	20.05.2020 16:13
4	jupyter-book - делаем интерактивную книгу из Jupyter Notebooks	<a href="http://github.com/executablebook">http://github.com/executablebook</a>		19.05.2020 11:14
5	Исключаем дефекты с изображения с помощью OpenCV	<a href="https://www.pyimagesearch.com/">https://www.pyimagesearch.com/</a>		20.05.2020 1:38

# Формат CSV

## Основное применение: выгрузки данных

- Для хранения больших объемов (до нескольких Гб) единообразных данных
- Можно читать и записывать файлы любого объема
- Самый компактный формат из всех
- Самый популярный формат для обмена данными у аналитиков
- Поддерживается MS Excel (эквивалентен плоской таблице)
- Поддерживает дозапись данных

### Минусы:

- Сложнее обрабатывать (нужны парсеры)
- Большой объём данных из-за мета-разметки (особенно XML)



# Формат CSV

## Десериализация в список

```
reader = csv.reader(file)
data = list(reader)
```

## Сериализация из списка

```
writer = csv.writer(file)
```

## Запись заголовка (только словарь)

```
writer.writerow()
```

## Десериализация в словарь

```
reader = csv.DictReader(file)
```

## Сериализация из словаря

```
writer = csv.DictWriter(file,
    fieldnames=reader.fieldnames)
```

## Запись данных (список, словарь)

```
writer.writerows(data)
writer.writerow(data)
```

## Настройки форматирования

```
csv.register_dialect()
```

# Формат CSV: Десериализация

```
7 ~ with open("files/sample.csv", newline="") as f:
→ 8     reader = csv.reader(f)
→ 9     rows_list = list(reader)
10
11 ~ with open("files/sample.csv", newline="") as f:
→ 12     reader = csv.reader(f)
→ 13 ~ for row in reader:
14         print(row[0])
15
16 ~ with open("files/sample.csv", newline="") as f:
→ 17     reader = csv.DictReader(f)
→ 18 ~ for row in reader:
19         print(row["title"])
```

# Формат CSV: Сериализация

```
29 # данные хранятся в news_list
30 ~ with open("files/sample2.csv", "w", newline="") as f:
→ 31     writer = csv.writer(f)
→ 32     writer.writerow(news_list[0]) # запись одной строки данных
→ 33     writer.writerows(news_list) # запись нескольких строк данных
```

# Формат CSV: Диалект

Настройки форматирования через `csv.register_dialect()`

```
delimiter=","  
quoting=csv.QUOTE_MINIMAL (QUOTE_ALL, QUOTE_NONNUMERIC, QUOTE_NONE)  
quotechar='"'  
escapechar='\\'
```

Сравните результат с `csv.QUOTE_MINIMAL` и `csv.QUOTE_ALL`:

- 5 Исключаем дефекты с изображения с помощью OpenCV,  
https://www.pyimagesearch.com/2020/05/18/image-inpainting-with-opencv-and-python/,,"Wed, 20 May  
2020 01:38:19 +0300"
  
- 5 "Исключаем дефекты с изображения с помощью OpenCV",  
"https://www.pyimagesearch.com/2020/05/18/image-inpainting-with-opencv-and-python/","",  
"Wed, 20 May 2020 01:38:19 +0300"

# Формат CSV: Диалект

```
→ 2 csv.register_dialect('customcsv', delimiter=',',  
    quoting=csv.QUOTE_MINIMAL, quotechar='"', escapechar='\\')  
3 ~ with open("files/sample.csv", "w", newline="") as f:  
→ 4     writer = csv.writer(f, dialect="customcsv")  
5     writer.writerows(some_data)
```

# Формат CSV: Важно

## Открытие файла

- указать кодировку `encoding="utf-8"` (`encoding="cp1251"`)
- убрать «лишний» перевод строки `newline=""`

## Чтение

- можно не указывать `"r"`

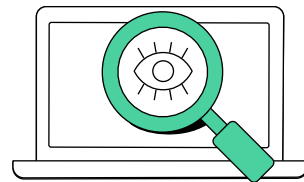
```
with open("files/sample.csv", encoding="utf-8", newline="") as f:
```

## Запись

- открыть для перезаписи `"w"`
- открыть для дозаписи (записи в конец файла)\* `"a"`

```
with open("files/sample.csv", "w", encoding="utf-8", newline="") as f:  
with open("files/sample.csv", "a", encoding="utf-8", newline="") as f:
```

# Демонстрация работы





**"Представьте, что вы сохраняете контакты из телефонной книги. Как бы выглядела строка CSV для одного контакта?"**

Напишите ваши ответы в чат.



# Формат JSON

3

# Формат JSON

```
1  {
2    "channel": {
3      "title": "Дайджест новостей о python",
4      "link": "https://pythondigest.ru/",
5      "description": "Русскоязычные анонсы свежих новостей о python и близлежащих технологиях.",
6      "lastBuildDate": "Wed, 20 May 2020 16:13:18 +0300",
7      "items": [
8        {
9          "title": "Как не править Python тесты",
10         "link": "https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&utm_medium=rss",
11         "description": "<p>И вынести тестируемые результаты вне кода. Это статья об автоматизации и увеличения удобства тестирования на Python.</p>",
12         "pubDate": "Wed, 20 May 2020 11:25:18 +0300"
13       }
14     ]
15   }
16 }
```

# Формат JSON

Основное применение: базы данных, выгрузки данных

- Для импорта/экспорта данных в базы данных (в т.ч. bson)
- Для сохранения вложенных структур данных
- Также применяется при передаче данных клиент<->сервер для сериализации иерархических объектов
- Самый популярный и простой в использовании формат для Python и Java программистов
- Является подмножеством формата YAML

## Минусы:

- Нельзя читать файл частично, только полностью

Документация по JSON <https://docs.python.org/3/library/json.html>

Онлайн редактор JSON <https://jsoneditoronline.org/>

# Формат JSON

## Десериализация

- Из файла: `json.load(file)`
- Из строки: `json.loads(str)`

## Сериализация

- В файл: `json.dump()`
- В строку: `json.dumps()`

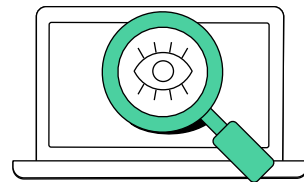
## Печать не-ascii символов, отступы

- `ensure_ascii=False, indent=2`

# Формат JSON: Десериализация и сериализация

```
28 import json
29 from pprint import pprint
30
31 data = {"channel": {"title": "Дайджест новостей о python",
32                    "link": "https://pythondigest.ru/"}}
33 json_str = '{"channel": {"title": "Дайджест новостей о python",
34                        "link": "https://pythondigest.ru/"}}'
35
36 with open("files/sample.json", encoding = "utf-8") as f:
37     json_data = json.load(f)
38     json_data = json.loads(json_str)
39     print(json_data)
40
41 with open("files/sample.json", "w") as f:
42     json.dump(dict_json, f, ensure_ascii=False, indent=2)
43     json_str = json.dumps(data, ensure_ascii=False, indent=2)
44     print(json_str)
```

# Демонстрация работы





**Как в JSON представить список любимых фильмов  
пользователя?**

Попробуйте за 1 минуту составить структуру и отправьте  
ваши ответы в чат.

# Формат YAML



4



# Формат YAML

```
1  channel:
2    description: Русскоязычные анонсы свежих новостей о python и близлежащих технологиях.
3    items:
4      - description: '<p>И вынести тестируемые результаты вне кода. Это статья об автоматизации
5        | и увеличения удобства тестирования на Python.</p>'
6        link: https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&
          utm_medium=rss
7        pubDate: Wed, 20 May 2020 11:25:18 +0300
8        title: Как не править Python тесты
9    lastBuildDate: Wed, 20 May 2020 16:13:18 +0300
10   link: https://pythondigest.ru/
11   title: Дайджест новостей о python
```

# Формат YAML

Основное применение: файлы конфигурации

- Самый компактный язык разметки
- Для создания файлов настроек
- Используется для описания классов, ресурсов и манифестов в API\*
- Может хранить несколько объектов в одном файле (в отличие от JSON)

## Минусы:

- Чувствительность к отступам – ошибки в пробелах/табах ломают структуру
- Ограниченная поддержка комментариев в многострочных строках – внутри | или > комментарии игнорируются

Официальный сайт YAML <https://yaml.org/>

Фреймворк обработки YAML для Python <https://github.com/yaml/pyyaml>

\*Swagger <https://editor.swagger.io/>

# Формат YAML

## Десериализация

- Из файла: `yaml.load(file)`, `yaml.load_all(file)*`
- Из строки: `yaml.loads(str)`

## Сериализация

- В файл: `yaml.dump`
- В строку: `yaml.dumps`

## Печать не-ascii символов, отступы

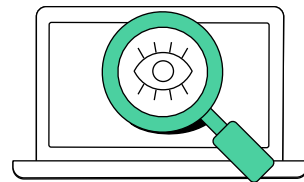
- `allow_unicode=True`, `default_flow_style=False`

\*Загрузка нескольких документов YAML <https://docs-python.ru/packages/modul-pyyaml-python/zagruzka-neskolkih-dokumentov-yaml/>

# Формат YAML: Десериализация и сериализация

```
50 import yaml
51 from pprint import pprint
52
53 data = {"channel": {"title": "Дайджест новостей о python",
54 | | | | | | | "link": "https://pythondigest.ru/"}}
55 with open("files/sample.yml", "w") as f:
→ 56 |     yaml.dump(data, f, allow_unicode=True, default_flow_style=False)
57
58 with open("files/sample.yml", encoding = "utf-8") as f:
→ 59 |     data = yaml.load(f, Loader=yaml.FullLoader)
60 |     pprint(data)
```

# Демонстрация работы





**Переведите этот JSON в YAML:**

```
{  
  "name": "Anna",  
  "age": 30,  
  "hobbies":  
    [ "reading", "travel" ]  
}
```

Напишите ваши ответы в чат.

# Формат XML

5



# Формат XML vs JSON

```
1 <?xml version="1.0" ?>
2 <rss xmlns:ns0="http://www.w3.org/2005/Atom" version="2.0">
3   <channel>
4     <title>Дайджест новостей о python</title>
5     <link>https://pythondigest.ru/</link>
6     <description>Русскоязычные анонсы свежих новостей о python и близлежащих
7     технологиях.</description>
8     <ns0:link href="https://pythondigest.ru/rss/" rel="self"/>
9     <language>ru-ru</language>
10    <lastBuildDate>Wed, 20 May 2020 16:13:18 +0300</lastBuildDate>
11    <item>
12      <title>Как не править Python тесты</title>
13      <link>https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&utm_medium=rss</link>
14      <description>&lt;p&gt;И вынести тестируемые результаты вне кода. Это
15      статья об автоматизации и увеличения удобства тестирования на Python.&
16      lt;p&gt;</description>
17      <pubDate>Wed, 20 May 2020 11:25:18 +0300</pubDate>
18      <guid>https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&utm_medium=rss</guid>
19    </item>
20    <item>
21      <title>Как построить диаграмму Венна с 50 кругами? Визуализация
22      множеств и история моего Python-проекта с открытым кодом</title>
23      <link>https://habr.com/ru/post/501924/?utm_campaign=501924&utm_source=habrahabr&utm_medium=rss</link>
24      <description>&lt;p&gt;Сегодня хочу рассказать вам про задачу
25      визуализации пересекающихся множеств. Поехали!&lt;p&gt;</description>
26      <pubDate>Wed, 20 May 2020 16:13:18 +0300</pubDate>
27      <guid>https://habr.com/ru/post/501924/?utm_campaign=501924&utm_source=habrahabr&utm_medium=rss</guid>
28    </item>
```

```
1 {
2   "channel": {
3     "title": "Дайджест новостей о python",
4     "link": "https://pythondigest.ru/",
5     "description": "Русскоязычные анонсы свежих новостей о ру
6     технологиях.",
7     "lastBuildDate": "Wed, 20 May 2020 16:13:18 +0300",
8     "items": [
9       {
10        "title": "Как не править Python тесты",
11        "link": "https://habr.com/ru/post/502278/?utm_campaign=502278&utm_source=habrahabr&utm_medium=rss",
12        "description": "<p&gt;И вынести тестируемые результаты
13        статья об автоматизации и увеличения удобства тестир
14        Python.</p&gt;",
15        "pubDate": "Wed, 20 May 2020 11:25:18 +0300"
16      },
17      {
18        "title": "Как построить диаграмму Венна с 50 кругами
19        множеств и история моего Python-проекта с открытым к
20        одом",
21        "link": "https://habr.com/ru/post/501924/?utm_campaign=501924&utm_source=habrahabr&utm_medium=rss",
22        "description": "<p&gt;Сегодня хочу рассказать вам про з
23        пересекающихся множеств. Поехали!</p&gt;",
24        "pubDate": "Wed, 20 May 2020 16:13:18 +0300"
25      }
26    ]
27  }
28 }
```



# Формат XML

Основное применение: сериализация объектов любой сложности

- Применяется при передаче данных клиент<->сервер для сериализации объектов
- Является стандартом обмена данными и сообщениями большинства информационных систем
- Применяется для хранения файлов конфигурации
- Большинство сложных форматов хранятся в XML (MS Office, OpenOffice, файлы разметки для нейросетей и пр.)

## Минусы:

- Избыточный синтаксис увеличивает объем данных и усложняет чтение
- Требуется парсер, медленнее JSON/YAML в чтении и записи, неудобен для ручного редактирования

# Формат XML

```
1 <rss xmlns:ns0="http://www.w3.org/2005/Atom" version="2.0">
2 <channel>
3   <title>Дайджест новостей о python</title>
4   <link>https://pythondigest.ru/</link>
5   <description>Русскоязычные анонсы свежих новостей о python и близлежащих
   технологиях.</description>
6 </channel></rss>
```

**Элемент:** <title>Дайджест новостей о python</title>

**Тег:** title

**Текст:** Дайджест новостей о python

**Атрибут:** version="2.0"

# Формат XML

```
1  <?xml version="1.0" ?>
Root 2  <rss xmlns:ns0="http://www.w3.org/2005/Atom" version="2.0">
      3  <channel>
4      <title>Дайджест новостей о python</title>
5      <link>https://pythondigest.ru/</link>
6      <description>Русскоязычные анонсы свежих новостей о pytho
      технологиях.</description>
7      <ns0:link href="https://pythondigest.ru/rss/" rel="self"
```

**Чтение дерева:** `xml.etree.ElementTree.parse()`

**Корень дерева:** `tree.getroot()`

`<rss xmlns:ns0="http://www.w3.org/2005/Atom" version="2.0">`

**Имя тега:** `root.tag`

**Текст внутри тега:** `root.text`

**Атрибуты тега:** `root.attrib`

**Сохранение:** `tree.write()`

# Формат XML: Работа с элементами

XPath – простой язык поиска по XML

**Поиск одного элемента:** `root.find(query)`

**Поиск нескольких элементов:** `root.findall(query)`

`query = XPath`

# XML: Загрузка из файла или словаря

```
75 import xml.etree.ElementTree as ET
76
77 # загрузка из файла
78 parser = ET.XMLParser(encoding="utf-8")
79 tree = ET.parse("files/newsافر.xml", parser)
80 root = tree.getroot()
81
82 # загрузка из словаря
83 from dicttoxml import dicttoxml
84 json_data = {"channel": {"title": "Дайджест новостей о python",
85                          "link": "https://pythondigest.ru/"}}
86 xml = dicttoxml(json_data) # результат - двоичная строка
87 root = ET.fromstring(xml.decode("utf-8"))
```

# XML: Работа с элементами

```
18 # получить корневой элемент дерева
19 root = tree.getroot()
20 # название тега (на примере корневого элемента)
21 print(root.tag)
22 # получение атрибутов тега
23 print(root.attrib)
24 # текст внутри тега
25 print(root.text)
26 # поиск элемента с помощью xpath
→ 27 xml_title = root.find("channel/title")
28 # текст внутри тега
29 print(xml_title.text)
30 # поиск всех элементов с помощью xpath
→ 31 xml_items = root.findall("channel/item")
32 print(len(xml_items))
33 for xmli in xml_items:
34     print(xmli.find("title").text)
```

# XML: Работа с элементами (JSON vs XML)

```
97 ~ json_data = {"channel": {"title": "Дайджест новостей о python",  
98     "link": "https://pythondigest.ru/"}}  
99 from dicttoxml import dicttoxml  
100 xml = dicttoxml(json_data)  
101 root = ET.fromstring(xml.decode("utf-8"))  
102  
→ 103 title_j = json_data["channel"]["title"]  
104 print(title_j)  
→ 105 title_x = root.find("channel/title")  
→ 106 print(title_x.text)
```

# XML: Сохранение

Первый способ. Методом write

```
1. import xml.etree.ElementTree as ET
2.
3. parser = ET.XMLParser(encoding="utf-8")
4. tree = ET.parse("files/newsافر.xml", parser)
5. root = tree.getroot()
6. tree.write("files/newsافر2.xml", encoding="utf-8")
```

Полный пример работы с XML <https://replit.com/@shorstko/netologyxmlfullsample>  
Самостоятельно: dict -> XML, XML -> dict <https://replit.com/@shorstko/netologyxmldictfullsample>

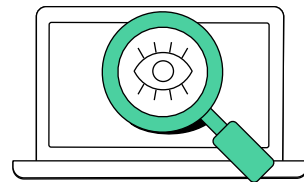


# XML: Сохранение

Первый способ. Методом write

```
1. import xml.etree.ElementTree as ET
2.
3. xml_str = '<root><channel type="dict"><title type="str">Дайджест новостей о python</title><link
   type="str">https://pythondigest.ru/</link></channel></root>'
4. root = ET.fromstring(xml_str)
5. tree = ET.ElementTree(root)
6. tree.write("files/sample1.xml", encoding="utf-8")
7.
8. import xmlformatter
9. # создаем formatter и задаем параметры форматирования
10. formatter = xmlformatter.Formatter(indent="2", indent_char=" ")
11. # форматируем. не забудьте указать кодировку!
12. prettyxml = formatter.format_string(ET.tostring(root)).decode("utf-8")
13. # записываем получившийся результат как текст
14. with open("files/sample2.xml", "w", encoding="utf-8") as f:
15.     f.write(prettyxml)
```

# Демонстрация работы





**Где вы чаще всего можете встретить XML в реальной жизни?**

Напишите ваши ответы в чат.

# Проблема кодировок



6

# Кодировки: проблемы с кириллицей

Один и тот же текст в кодировке Windows 1251 и utf-8

Windows 1251 (один байт на букву)

00000000:	C8 F1 F2 EE F0 E8 FF 20	F3 20 EC E5 ED FF 20 F1		История у меня с
00000010:	EB E5 E4 F3 FE F9 E0 FF	3A 20 EF EE E7 ED E0 EA		ледующая: позн
00000020:	EE EC E8 EB F1 FF 20 F1	20 E4 E5 E2 F3 F8 EA EE		омился с девушко
00000030:	E9 20 E8 E7 20 F1 E2 EE	E5 E3 EE 20 E8 ED F1 F2		й из своего инст
00000040:	E8 F2 F3 F2 E0 2E 0D 0A	D1 ED E0 F7 E0 EB E0 20		итута...Сначала

utf-8 (два байта на букву)

00000000:	EF BB BF	D0 98 D1 81 D1	82 D0 BE D1 80 D0 B8 D1		п»ïP.cŕC,PſCŧPëC
00000010:	8F 20 D1 83 20 D0 BC D0	B5 D0 BD D1 8F 20 D1 81		Ů Cŕ PjPµPSCŮ Cŕ	
00000020:	D0 BB D0 B5 D0 B4 D1 83	D1 8E D1 89 D0 B0 D1 8F		P»PµPrCŕCŧCŹP°CŮ	
00000030:	3A 20 D0 BF D0 BE D0 B7	D0 BD D0 B0 D0 BA D0 BE		: PïPſP-PſP°PëPſ	
00000040:	D0 BC D0 B8 D0 BB D1 81	D1 8F 20 D1 81 20 D0 B4		PjPëP»CŕCŮ Cŕ Pr	

# Кодировки: проблемы с кириллицей

Ожидания при сохранении в JSON:

```
2  "channel": {
3    "title": "Дайджест новостей о python",
4    "link": "https://pythondigest.ru/",
5    "description": "Русскоязычные анонсы свежих новостей о python и близлежащих
технологиях.",
6    "lastBuildDate": "Wed, 20 May 2020 16:13:18 +0300",
```

Результат при сохранении в JSON:

```
2  "channel": {
3    "title": "\u0414\u0430\u0439\u0434\u0436\u0435\u0441\u0442 \u043d\u043e\u0432\u043e\u0441\u0442\u0435\u0439 \u043e \u043f\u044b\u0442\u0438\u043e\u043d",
4    "link": "https://pythondigest.ru/",
5    "description":
"\u0420\u0443\u0441\u043a\u043e\u044e\u0437\u044b\u0447\u043d\u044b\u0435 \u0430\u043d\u043e\u043d\u0441\u044b \u0441\u0432\u0435\u0436\u0438\u0445 \u043d\u043e\u0432\u043e\u0441\u0442\u0435\u0439 \u043e \u043f\u044b\u0442\u0438\u043e\u043d \u0438 \u0431\u043b\u0438\u0437\u0435\u0436\u0430\u0449\u0438\u043c \u0442\u0435\u0445\u043d\u043e\u043b\u043e\u0433\u0438\u044f\u043c.",
6    "lastBuildDate": "Wed, 20 May 2020 16:13:18 +0300",
```

Ошибка: `json.dump(dict_json, f, ensure_ascii=False, indent=2)`

# Кодировки: проблемы с кириллицей

Для решения проблемы явно указывайте кодировку в функциях чтения и записи:

```
59 with open("files/sample.json", encoding = "utf-8") as f:  
60     data = json.load(f)
```



Результат:

```
2     "channel": {  
3         "title": "Дайджест новостей о python",  
4         "link": "https://pythondigest.ru/",  
5         "description": "Русскоязычные анонсы свежих новостей о python и близлежащих  
        технологиях.",  
6         "lastBuildDate": "Wed, 20 May 2020 16:13:18 +0300",
```

# Итоги и ваши вопросы

7



# Сегодня мы

- Познакомились с различными форматами данных:
  - Плоскими - CSV
  - Древовидными - JSON, YAML, XML
  
- Разобрались, что такое сериализация и десериализация
  
- Научились работать со всеми представленными форматами:
  - Читать их из файла или строк
  - Искать необходимую информацию в данных
  - Редактировать данные
  - Записывать их в файл или в строку





**Ваши вопросы?**

# Работа с разными форматами данных

Елена Никитина

Руководитель проектов ООО «Аналитические программные решения»

