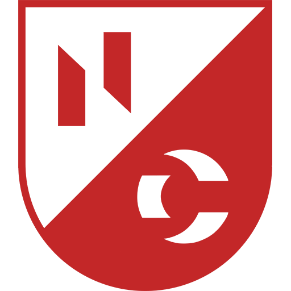
**Некоммерческое образовательное учреждение  
Учебно-научно-производственный комплекс**

**«Международный Университет Кыргызстана»**

**Среднее профессиональное образование**

**«Nomad» Колледж**



Программа «Программирование в компьютерных системах»

**Курманалиева Адисса Аралбаевна**

**КУРСОВАЯ РАБОТА**

*На тему:* «Работа с XML и JSON в Python*»*

Руководитель курсовой работы:

преп.

\_\_\_\_\_\_\_\_\_\_\_ Уралбек уулу С.

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

**Бишкек, 2024 г.**

**СОДЕРЖАНИЕ:**

[**ВВЕДЕНИЕ** 3](#_Toc156167033)

[**ГЛАВА 1. Введение в XML и JSON** 5](#_Toc156167034)

[**1.1. Общее представление о XML и JSON** 5](#_Toc156167035)

[**1.2. Основные принципы организации данных в этих форматах** 5](#_Toc156167036)

[**ГЛАВА 2. Модуль работы с XML и JSON в Python** 8](#_Toc156167037)

[**2.1. Модули для работы с XML** 8](#_Toc156167038)

[**2.2. Модули для работы с JSON** 10](#_Toc156167039)

[**ГЛАВА 3. Преобразование между XML и JSON** 12](#_Toc156167040)

[**3.1. Методы преобразования данных между XML и JSON в Python** 12](#_Toc156167041)

[**3.2. Практическое применение преобразования** 14](#_Toc156167042)

[**ГЛАВА 4. Применение в реальных проектах** 16](#_Toc156167043)

[**4.1. Пример и его описание, преимущества и недостатки** 16](#_Toc156167044)

[**ГЛАВА 5. Сравнение и выбор формата** 20](#_Toc156167045)

[**5.1. Что выбрать между XML и JSON в разных ситуациях** 20](#_Toc156167046)

[**ГЛАВА 6. Безопасность при работе с данными** 22](#_Toc156167047)

[**6.1.** **Как защитить программы от возможных атак** 22](#_Toc156167048)

[**6.2.** **При обработке XML** 22](#_Toc156167049)

[**6.3.** **При обработке JSON** 22](#_Toc156167050)

[**ЗАКЛЮЧЕНИЕ** 24](#_Toc156167051)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 25](#_Toc156167052)

# **ВВЕДЕНИЕ**

**Актуальность темы.** Тема «Работа с XML и JSON в Python» является весьма актуальной в сфере программирования, привлекая к себе внимание разработчиков и специалистов в области информационных технологий. XML и JSON, как форматы данных широко используются для передачи и хранения данных в структурированной форме, в сфере веб-разработки, обмена данными между приложениями, а также в создании конфигурационных файлов. Python, благодаря свой популярности и эффективности, как один из языков программирования, предоставляет разработчикам удобные инструменты для работы с обоими форматами, что позволяет этой теме быть востребованной и актуальной для разработчиков. В контексте различных форматов данных, обширных объемов данных и их тесного взаимодействия. Навык работать с XML и JSON в Python остается неотъемлемым элементом для эффективной разработки и поддержки в сфере современных программ, учитывая условия постоянной эволюции технологий и прогрессирующего увеличения объемов, структурированных данных.

**Цель работы**.

Целью работы является исследование и освоение методов работы с форматами данных XML и JSON в языке программирования Python.

Для достижения цели поставлены и решены следующие **задачи:**

**-** Изучение основных концепций и структур данных в форматах XML и JSON.

**-** Освоение библиотек и модулей Python для работы с XML и JSON.

**-** Разработка примеров кода для чтения, записи, изменения и создания XML- и JSON-данных.

- Анализ использования XML и JSON в практических сценариях, таких как веб-разработка, обмен данными между приложениями.

- Проведение сравнительного анализа преимуществ и недостатков использования XML и JSON в различных сферах программирования.

- Рассмотрение вопросов безопасности при обработке XML и JSON, включая защиту от потенциальных угроз.

- Формирование рекомендаций по выбору формата данных в зависимости от конкретных задач и требований проекта.

**Объектом исследования** является процесс работы с форматами данных XML и JSON в языке программирования Python.

**Предметом исследования** является методы и техники обработки данных в форматах XML и JSON при использовании языка программирования Python.

**Практическая значимость исследования** проявляется в повышении профессиональных компетенций разработчиков и IT-специалистов. Это практически полезно, поскольку современные проекты все чаще сталкиваются с потребностью эффективного обращения и обработки структурированных данных. Полученные знания в ходе исследования также могут предоставить возможность оптимизировать процессы обработки данных. Наконец, применение этих навыков в реальных проектах, вносит вклад в повышении и эффективности, а также в успешную реализацию программных продуктов.

**Структура работы** соответствует цели и задачам исследования. Работа состоит из введения, шести глав (по одному до трех парагрофов в каждой), заключения и списка использованной литературы.

# **ГЛАВА 1. Введение в XML и JSON**

## **1.1. Общее представление о XML и JSON**

XML (eXtensible Markup Language) – это расширяемый язык разметки, предназначенный для хранения и передачи данных в структурированной форме. Благодаря своему расширяемому формату, позволяет определить свои правила разметки данных. XML чаще используется для обмена информацией между программами через интернет пространство, а также для хранения данных в файловой системе.

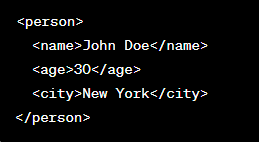
JSON (JavaScript Object Notation), является легковесным и понятным для человека форматом обмена данным, и его простота в применении делает его широко используемым в современных веб-приложениях и API.

# **1.2. Основные принципы организации данных в этих форматах**

Оба формата данных, XML (eXtensible Markup Language) и JSON (JavaScript Object Notation), являют собой способ организации и передачи структурированных данных, но каждый из них имеет свои уникальные особенности и принципы организации информации.

**XML (eXtensible Markup Language):**

XML представляет данные в виде дерева, для этого используются теги для определения структуры данных. Пример:



Основные характеристики:

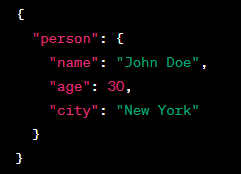
- Теги: Элементы данных заключены в теги и выделенными угловыми скобками, которые определяют начало и конец элемента. Например: '<name>John Doe </name>'.

- Атрибуты: Элементы могут содержать атрибуты, для дополнительной информации в виде пар «ключ-значение». Пример: '<person gender=''male''> John Doe </person>'.

- Вложенность: Элементы могут быть вложены друг в друга, что позволяет создать иерархию между данными.

**JSON (JavaScript Object Notation):**

JSON представляет данные в виде пар «ключ-значение» и поддерживает вложенные структуры. Пример:



Основные характеристики JSON:

- Объекты: Данные представлены в виде объектов, содержащих пары «ключ-значение», например, «''name'': ''John Doe''».

- Массивы: Массивы представляют упорядоченные списки значений.

- Простые типы данных: JSON поддерживает строки, числа, логические значения (true/false), null и другие структуры данных.

Общие черты между XML и JSON:

- Читаемость: Оба формата ориентированы на читаемость и поддерживают разбор человека.

- Независимость от платформы: XML и JSON являются независимыми от платформы форматами данных и могут быть использованы на различных языках программирования.

Выбор между XML и JSON зависит от сферы применения, требований проекта и личных предпочтений разработчиков.

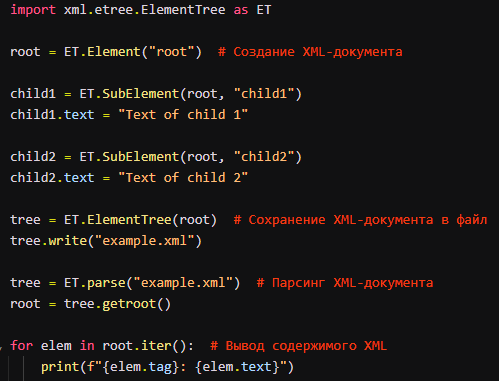
# **ГЛАВА 2. Модуль работы с XML и JSON в Python**

В языке программирования Python существует несколько встроенных модулей для работы с XML и JSON, предоставляющие удобные средства для чтения, записи и обработки данных в этих форматах.

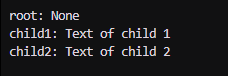
## **2.1. Модули для работы с XML**

1. ElementTree:

Этот модуль входит в стандартную библиотеку Python и предоставляет легкий способ обработки XML. Пример использования:



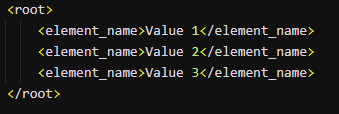
И его вывод, будет выглядеть так:



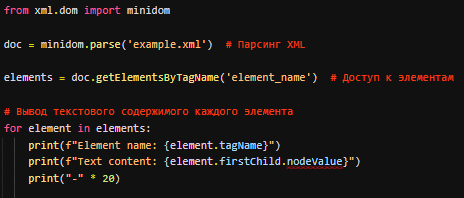
2. minidom:

Этот модуль также является частью стандартно библиотеки и предоставляет DOM-подход для работы с XML.

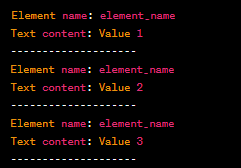
Пример использования: предположим, что XML-документ выглядит следующим образом:



Для вывода текстового содержимого каждого элемента продолжим его код:



Выше указанный код пройдется по всем элементам с тегом 'element\_name'. Он выведет сначала имя элемента, а затем и его текстовое содержание. Вывод будет выглядеть следующим образом:



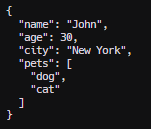
## **2.2. Модули для работы с JSON**

1. json:

«json», является встроенным модулем для работы с JSON. Пример использования:

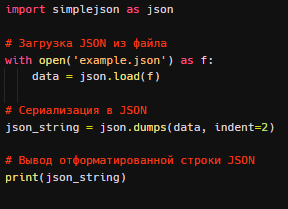


Вывод кода будет следующим:

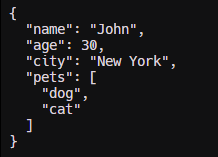


2. simplejson:

Сторонняя встроенная библиотека, которая является улучшенной версией встроенного модуля «json». Устанавливается с помощью команды 'pip install simplejson'. Пример использования:



После выполнения кода, будет выведена отформатированная строка JSON с отступами в 2 пробела:



# **ГЛАВА 3. Преобразование между XML и JSON**

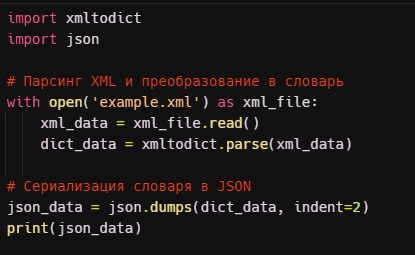
## **3.1. Методы преобразования данных между XML и JSON в Python**

В Python существует много разных методов для преобразования данных между XML и JSON. Ниже приведены два основных подхода:

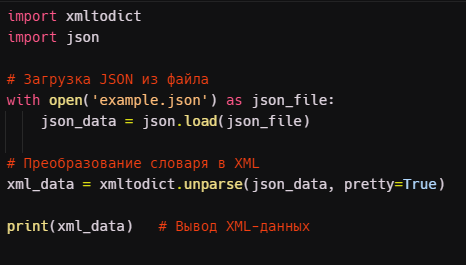
1. Использование библиотек:

Большее количество сторонних библиотек облегчают способ преобразования между XML и JSON. Два часто используемых библиотек, это: «xmltodict», «json» (встроенная).

Пример преобразования из XML в JSON:



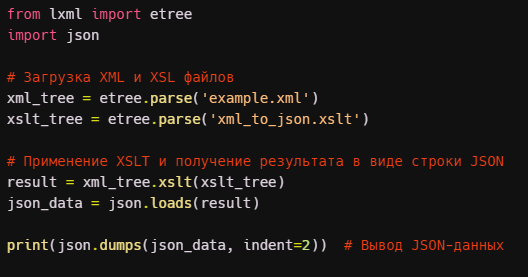
Пример преобразования из JSON в XML:



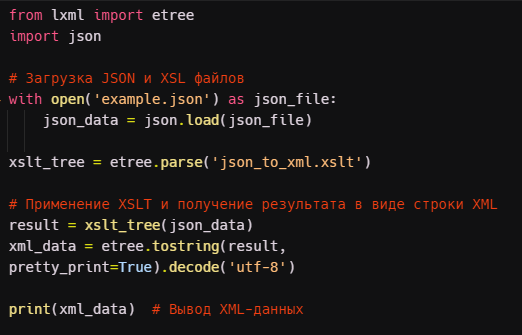
1. Использование XSLT:

Один из других методов – использование таблиц стилей XSLT. Этот метод требует написания таблиц стиле XSLT для каждого направления преобразования.

Пример преобразования из XML в JSON с использованием XSLT:



Пример преобразования из JSON в XML с использованием XSLT:



Оба метода имеют свои преимущества и недостатки. Как говорилось ранее выбор зависит от предпочтения разработчиков и его конкретно сферы применения.

## **3.2. Практическое применение преобразования**

Практическое применения преобразования данных между XML и JSON, может быть полезным в различных сценариях, где требуется совместимость между системами, работающими с разными структурами данных. Рассмотрим несколько сценариев, когда такие преобразования могут быть полезными.

1. Интеграция между системами:

Благодаря преобразования данных облегчается процесс обмен информацией между системами, использующие разные форматы данных, что может значительно повысить эффективность и сделать легким процесс работы.

1. Обмен данным в веб-приложениях:

В веб-разработке, где данные могут передаваться как XML или JSON. Преобразование может быть ценным для соответствия ожиданиям клиента и формату данных на сервере.

1. Обработка данных в различных языках программирования:

Языки программирования могут предпочитать работать с данными в разных форматах, так, например, Python может легко взаимодействовать и обрабатывать JSON. По это причине преобразование, может быть полезным при интеграции компонентов, написанных на разных языках программирования.

1. Работа с API:

Веб-сервисы предоставляют данные в JSON или XML, преобразование между этими форматами может быть необходимо при работе с различными API.

1. Конвертация данных для визуализации:

Во время процесса работы с инструментами для анализа или визуализации данных, которые могут предпочитать один формат данных перед другим. Преобразование может помочь для подготовки данных.

1. Адаптация к стандартам обмена данными:

В контексте стандартов электронной коммерции или других областей, где применяются определенные форматы. Преобразование позволяет соответствовать стандартам и обеспечивать совместимость.

В целом, способность преобразования между XML и JSON предоставляет гибкость и универсальность в различных сценариях обмена и обработки данных.

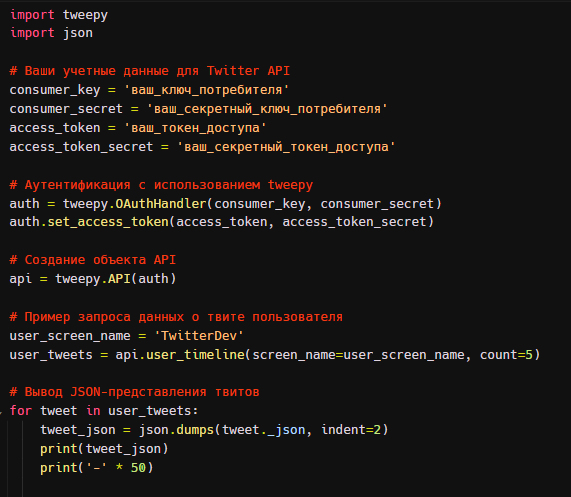
# **ГЛАВА 4. Применение в реальных проектах**

## **4.1. Пример и его описание, преимущества и недостатки**

Работа с XML и JSON в Python не редко эксплуатируется в реальных проектах. Вот пример его использования.

1. Интеграция с Twitter/X API:

Этот пример демонстрирует использование библиотеки 'tweepy' на языке Python для взаимодействия с Twitter API. Полученные данные о твитах пользователя TwitterDev выводятся в формате JSON.



Преимущества:

- Простата взаимодействия, благодаря использованию библиотеки 'tweepy' взаимодействие с Twitter API стало легче.

- Универсальность и гибкость, позволяет упростить обработку полученных данных и их интеграцию в другие части приложения.

- Гибкость и читаемость кода. По причине использования JSON-представления, можно легко изменять и адаптировать обработку данных при его необходимости.

Недостаток:

- Ограничение в объеме данных. Запрос ограничен только последними пятью твитами пользователя для цели примера. В существующем же приложении может потребоваться обработка огромного объема данных.

- Ограничения API. Twitter API может иметь ограничения по количеству запросов в единицу времени, что может принудить иметь учета в проекте.

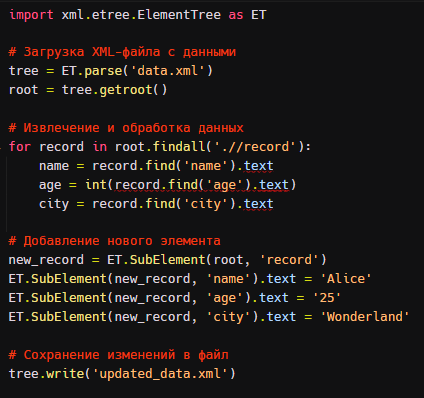
- Безопасность данных. В рассматриваем примере не предусмотрены меры безопасности, такие как шифрование или обработка конфиденциальных данных. В реальном же приложении нужно и важно обеспечивать безопасность передаваемой информации.

- Обновление ключей и токенов. Учетные данные для Twitter API (ключи и токены) могут изменяться, и приложение должно быть обновлено соответственно.

Вывод: Пример демонстрирует легки способ для получения данных из Twitter API и их представления в удобном формате для обработки данных в формате JSON. Однако стоит учитывать, что для создания настоящего приложения стоит учитывать все нюансы.

1. Работа с XML в проекте для анализа данных.

Представим, у нас есть проект, который занимается анализом данных, хранящихся в формате XML. Данные предоставляют собой записи о людях, где каждая запись содержит информацию о имени, возрасте и городе проживания.



Объяснение:

1. Загрузка данных. Сначала мы воспользуемся библиотекой «xml.etree.ElementTree» для скачивания данных их XML-файла.
2. Извлечение и обработка данных. Пользователь проходит по каждой записи в XML, извлекая информацию о имени, возрасте и городе, и проводим дополнительную обработку.
3. Добавление ново записи. Для илюстрации добавления данных в XML, мы создаем новую запись 'Alice' и добавляем ее в структуру данных.
4. Сохранение изменений. На этом этапе, пользователь сохраняет обновленные даты в тот же XML-файл.

Использование:

Вышеуказанный код может стать частью более масштабного проекта для анализа и обработки данных. К примеру, в медицинском проекте это могли бы быть записи медицинские о больных, а в образовательном проекте могут быть – данные о студентах.

# **ГЛАВА 5. Сравнение и выбор формата**

## **5.1. Что выбрать между XML и JSON в разных ситуациях**

Выбор между XML и JSON зависит от конкретно поставленной задачи вашего проекта, структуры данных, удобства чтения и много других факторов. Вот несколько рекомендации для выбора между XML и JSON в различных ситуациях.

Используйте XML, если:

- Сложная структура данных: XML имеет более мощны механизм представления структурированных данных с помощью вложенных в себе элементов и атрибутов, что может быть очень полезным при работе с сложными структурами данных.

- Документно-ориентированные данные: XML часто применяется для представления документно-ориентированных данных, таких как статьи, файлы и т.д.

- Необходимость поддержки схем данных (XSD): Если вам потребуется определить схему данных и проводить над ними валидацию, то XML, с поддержкой XSD, будет вашим выбором.

Используйте JSON, если:

- Нужна простата чтения и записи: JSON по сравнению с XML имеет компактны и легко читаемый синтаксис, что делает его легким для написания и чтения человеком.

- Структурированные данные для веб-приложений: JSON применяется в веб-приложениях, а также он легко интегрируется с JavaScript.

- Данные API: Многие веб-серверы представляют данные в формате JSON для простоты взаимодействия через API.

- Программирование в Python, JavaScript и др.: Благодаря поддержке в большинстве языков программирования, обмен происходит удобно между различными компонентами системы.

# **ГЛАВА 6. Безопасность при работе с данными**

* 1. **Как защитить программы от возможных атак**

Обработка данных в форматах XML и JSON может представлять угрозы безопасности, такие как атаки на внедрение кода (к примеру, инъекции) или же атаки на отказ в обслуживании (DoS). Для обеспечения безопасность от возможных атак, во время процесса обработки данных, следует придерживаться ряда мер предосторожностей.

* 1. **При обработке XML**

- Избегайте внедрение кода (XXE). Ограничьте или отключите полностью его обработку внешних сущностей XML, для своевременного предотвращения возможно атаки, связанные с внедрением сущностей.

- Валидация данных. Проводите валидацию данных по схеме XML (XSD). Это поможет предотвратить атаки, связанные с неверным форматом данных.

- Применяйте безопасные библиотеки парсинга. Пользуйтесь только надежными и безопасными библиотеками парсинга XML, такие как «ElementTree», которая является стандартно библиотекой в Python и «lxml». Также не пользуйтесь во время выполнения разбора встроенных функций строк.

* 1. **При обработке JSON**

- Используйте безопасные библиотеки. Используйте проверенные библиотеки для работы с JSON. В стандартной библиотеке Python есть встроенный модуль «json», предоставляющий безопасные функции для сериализации и десериализации.

- Ограничение вложенности и объема данных. Ограничьте максимальную возможность вложенности и размера данных JSON, для предотвращения атаки на отказ в обслуживании (DoS) с использованием больших и бесконечно вложенных структур данных.

- Избегайте атак внедрения кода. Избегайте выполнения кода, получаемого из JSON-данных.

# **ЗАКЛЮЧЕНИЕ**

Работа с XML и JSON в Python предполагает собой немаловажный фактор нынешней реализации программного обеспечения. Оба формата данных обширно используются для взаимообмена информацией между разнообразнейшими системами и приложениями, снабжая универсальные и гибкие средства передачи и хранения данных.

XML, с его расширенными перспективами для предоставления сложных структур данных, часто используется в сферах, где важна читаемость и структурирование информации. С другой стороны, JSON, с достаточно компактным синтаксисом и легкочитаемостью, стал основополагающим форматом для обмена данными в веб-разработке и API.

Python владеет мощными инструментами для работы с обоими форматами: модули «xml.etree.ElementTree» и «xml.dom» для XML, а также подключенные модуль «json» для работы с JSON. Эти инструменты снабжают непринужденность использования и высокую производительность при обработке данных в надлежащих форматах.

Однако, при всей из удобности, необходимо обусловливать вопросы, связанные с защищенностью при обработке данных в форматах XML и JSON. Систематические проверки входных данных, использование безопасных библиотек парсинга, регулирование размеров и вложенности данных, а также внимания к деталям гарантируют безотказную защищенность от предполагаемых угроз.

Работа с XML и JSON в Python не только гарантирует оптимальный обмен данными, но также акцентирует значительность баланса между функциональностью и безопасностью в нынешней разработке программного обеспечения.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Официальная документация Python
2. XML External Entity (XXE) Processing / Ресурс OWASP
3. JSON Security Cheat Sheet / Ресурс OWASP
4. W3C XML Tutorial / W3Schools
5. JSON.org / JSON