Geekbrains

«Разработка веб-скрапера для извлечения данных с веб-сайтов с использованием библиотек Beautiful Soup, requests и pandas»

IT-специалист:

Программист Python

Сенченков К.О.

Санкт-Петербург

2024

Оглавление:

1. Введение:

В открытом доступе интернета находиться огромное количество информации. Чтобы обработать и правильно использовать эту информацию простому человеку приходить тратить большое количество времени. Чтобы избежать данной проблемы и более сосредоточенно сфокусировать свое внимание на требуемых задачах можно обраться к скрайпингу.

Скрайпинг – представляет собой процесс извлечения требуемой информации и данных с web ресурсов.

Существует несколько разновидностей веб-скрапинга, основными из которых являются:

1. Traditional Web Scraping (традиционный веб-скрапинг): Этот подход включает в себя написание собственных скриптов или программ для извлечения данных с веб-страниц. Обычно он основан на использовании инструментов, таких как BeautifulSoup, Scrapy, или Selenium в различных языках программирования, например, Python или JavaScript.

2. Automatic Data Collection (автоматическое сбор данных): В этом случае используются специализированные программы или службы, которые предоставляют API для извлечения данных с веб-сайтов или установления автоматического сбора информации.

3. Monitor and Extract (мониторинг и извлечение): Этот тип веб-скрапинга включает непрерывный мониторинг изменения веб-страниц и автоматическое извлечение данных после обновлений или изменений.

4. Комбинированный подход: Включает в себя комбинацию традиционного веб-скрапинга с использованием API, а также других методов автоматического сбора данных для получения наиболее полной информации.

Каждый из этих методов имеет свои преимущества и недостатки, и эффективный выбор зависит от требований конкретного проекта и доступности данных.

Различные организации и отдельные пользователи могут использовать разновидности веб-скрапинга для получения информации.

Исследователи и академические учреждения используют веб-скрапинг для извлечения данных для исследования, аналитических целей и создания базы данных для академических публикаций. Коммерческие организации используют веб-скрапинг для анализа конкурентной информации, ценовой политики или для мониторинга обратной связи от клиентов. Финансовые учреждения используют автоматизированные методы веб-скрапинга для мониторинга финансовых показателей, сбора данных о рынке и анализа трендов. Журналисты и СМИ используют скрапинг для сбора информации о событиях, общественном мнении и других связанных данных. Многие компании в области информационных технологий и разработки используют веб-скрапинг для получения данных для обучения моделей машинного обучения или других технологических приложений. Эксперты по маркетингу и рекламе используют скрапинг для анализа данных из социальных сетей, отзывов клиентов и других маркетинговых данных.

Таким образом, разновидности веб-скрапинга применяются в различных сферах для получения, анализа и использования данных в целях исследования, бизнеса, информационных технологий и других областей.

Существует несколько особенностей веб-скрапинга, на которые стоит обратить внимание:

1. Юридические и этические вопросы: Некоторые веб-сайты запрещают скрапинг данных в своих условиях использования. Важно следить за соблюдением законодательства об авторских правах и правах собственности на данные при проведении веб-скрапинга.

2. Безопасность и защита данных: При скрапинге данных можно столкнуться с риском нарушения безопасности или утечки конфиденциальной информации. Важно принимать меры для защиты данных, с которыми вы работаете, а именно:

2.1. Получение разрешения: Удостоверьтесь, что у вас есть законное право скрапить данные с веб-сайтов, так как незаконный доступ к данным может повлечь за собой правовые последствия.

2.2. Использование прокси-серверов: Для сокрытия вашего реального IP-адреса и предотвращения блокировок со стороны серверов скрапинга данных можно использовать прокси-серверы.

2.3. Ограничение доступа: Обеспечьте ограниченный доступ к системе скрапинга данных, чтобы предотвратить несанкционированное использование.

2.4. Обработка персональных данных: Если данные содержат персональную информацию, убедитесь, что вы соблюдаете требования по защите персональных данных, установленные соответствующими законами.

2.5. Шифрование: Зашифруйте хранимые и передаваемые данные, чтобы предотвратить их несанкционированный доступ.

2.6. Мониторинг и обновление безопасности: Регулярно мониторьте систему скрапинга данных на наличие уязвимостей и обновляйте защитные меры.

3. Изменчивость веб-страниц: Веб-страницы могут меняться со временем, что может привести к неправильному извлечению данных. Постоянное обновление скриптов скрапинга может быть необходимо для поддержания текущей работы.

4. Ограничения доступа к данным: Некоторые веб-сайты могут вводить ограничения на доступ к данным для скрапинга, такие как капчи, ограничение скорости запросов и другие технические меры.

5. Политический и моральный аспект: Использование данных, собранных путем скрапинга, может вызвать разногласия или привести к негативным мнениям, особенно если скрапинг данных осуществляется без согласия владельца источника.

При проведении веб-скрапинга важно учитывать эти особенности и тщательно оценивать потенциальные риски и негативные последствия, а также соблюдать законные и этические нормы.

Однако, не смотря на все вышеизложенные аспекты именно этот способ помогает нам максимально и эффективно использовать время для достижения поставленных задач.

Сайты могут устанавливать защиту от веб-скрапинга по нескольким причинам:

1. Защита от перегрузки сервера: Процессы веб-скрапинга могут создавать высокую нагрузку на сервер, что может привести к снижению производительности и доступности сайта для обычных пользователей.

2. Защита от потери контента: Некоторые владельцы сайтов опасаются, что веб-скрапинг может привести к копированию и публикации их контента без разрешения, что может ущемлять их авторские права и результаты труда.

3. Защита от сбора данных: В некоторых случаях, сайты могут стремиться защитить свои данные от сбора конкурирующими компаниями или сторонними лицами.

4. Соблюдение правил доступа: Некоторые сайты предоставляют открытый доступ к своим данным через API или другие специализированные интерфейсы, и они могут хотеть, чтобы внешние приложения использовали именно эти методы доступа, а не веб-скрапинг.

Вопрос о законности использования веб-скрапинга сайтов зависит от конкретных обстоятельств и специфики данной деятельности. В некоторых случаях, веб-скрапинг может быть допустимым, если соответствует нормам законодательства и правилам использования информации, предоставленной веб-сайтом. Однако, некоторые веб-сайты могут запрещать веб-скрапинг в своих условиях использования или нарушать авторские права, тогда использование скрапинга без разрешения может быть незаконным.

Важно учитывать такие факторы, как цели скрапинга, тип собираемой информации, способы использования и возможные нарушения частных прав, а также особенности местного законодательства. В целях безопасности и легальности рекомендуется всегда ознакамливаться с правилами использования

сайта и консультироваться с юристом или специалистом в области информационной безопасности при создании программ для веб-скрапинга.

Основные цели скрапинга включают в себя:

1. Анализ данных: Получение структурированных данных с веб-сайтов для анализа, изучения рынка, трендов, прогнозирования и принятия бизнес-решений.

2. Мониторинг изменений: Отслеживание изменений в ценах, оценках, рейтингах, обзорах и других параметрах, позволяет оставаться в курсе событий, изменений рынка и конкуренции.

3. Сравнительный анализ: Сбор данных для сравнительного анализа продуктов, услуг, курсов и других параметров, помогающий в принятии обоснованных решений.

4. Генерация контента: Использование данных для создания информационных ресурсов, агрегаторов и других приложений.

5. Прогнозирование и оптимизация: Использование скрапинга для прогнозирования спроса, управления запасами, планирования рекламных кампаний и оптимизации бизнес-процессов.

6. Исследования и обучения: Получение данных для научных исследований, учебных целей и обогащения предметной области.

1.1 Формулирование проблемы

В наше время все больше данных находится в открытом доступе на веб-сайтах, однако, их извлечение и преобразование в удобочитаемый формат становится значительной проблемой из-за разнообразия структур веб-страниц. Ручной сбор данных не только трудоемок, но и подвержен ошибкам, что актуализирует необходимость разработки автоматизированного подхода к извлечению информации с веб-сайтов. Процесс автоматического сбора данных, или веб-скрапинг, выделяется своей важностью в контексте обеспечения точности собранных данных, сокращения времени и ресурсов, потраченных на

этот процесс, а также возможности проведения аналитики и исследований на основе полученных данных.

В этом контексте, разработка веб-скрапера с использованием библиотек Beautiful Soup и requests представляет собой важную задачу, направленную на упрощение извлечения данных с веб-сайтов. Использование указанных технологий позволяет создать эффективный инструмент для автоматизации процесса сбора данных, а также обеспечивает возможности дальнейшего анализа и обработки полученной информации. Путем разработки данного веб-скрапера мы можем значительно улучшить эффективность и точность извлечения данных, обеспечивая ценную информацию для дальнейших аналитических и исследовательских задач.

1.2 Цель проекта

Целью данного проекта является разработка эффективного веб-скрапера, способного извлекать структурированные данные с веб-сайтов, основываясь на библиотеках Beautiful Soup и requests. Веб-скрапинг - это мощный инструмент для автоматизации процесса сбора данных, однако задача разработки высокоэффективного и надежного скрапера извлечения информации с веб-сайтов не является тривиальной. Целью данной работы является создание надежного инструмента, способного предоставлять пользователю структурированные данные в удобочитаемой форме, обеспечивая высокую точность и скорость извлечения информации.

Кроме того, целью проекта также является изучение и понимание принципов работы библиотек Beautiful Soup и requests, применение их функционала для сбора данных с веб-сайтов, а также оптимизация процесса скрапинга для обеспечения высокой производительности.

Оптимизация процесса скрапинга веб-сайтов может быть достигнута путем применения следующих стратегий:

1.Выбор правильных инструментов: Использование специализированных библиотек и инструментов, таких как Beautiful Soup, Scrapy, или Puppeteer, которые позволяют эффективно извлекать данные из веб-страниц.

2. Кэширование: Сохранение результатов предыдущих запросов и страниц может помочь избежать избыточных запросов к серверу и ускорить процесс скрапинга.

3. Асинхронные запросы: Использование асинхронных HTTP-запросов и многопоточности может ускорить скорость скрапинга путем одновременной обработки нескольких запросов.

4. Использование API: Проверьте, есть ли у сайта API, через который можно получить требуемую информацию. Использование API может быть более надежным и эффективным способом получения данных, чем скрапинг HTML-кода.

5. Обработка ошибок: Реализуйте механизмы обработки ошибок, чтобы скрипт скрапинга мог автоматически обрабатывать прерывания, временные ошибки и другие проблемы, которые могут возникнуть в процессе скрапинга.

6. Мониторинг и логирование: Добавьте функционал мониторинга и логирования, чтобы отслеживать процесс скрапинга, выявлять проблемы и оптимизировать производительность.

7. Учет частотных ограничений: Некоторые веб-сайты могут иметь ограничения на количество запросов в единицу времени. Учитывайте эти ограничения и регулируйте скорость запросов соответственно.

Применение этих методов может повысить эффективность и производительность процесса скрапинга веб-сайтов.

В результате выполнения и применения методов оптимизации ожидается разработка инструмента, который будет способен эффективно и точно извлекать данные из веб-сайтов, обеспечивая пользователями необходимую информацию для последующего анализа, обработки и использования в различных областях, включая исследования, бизнес-аналитику, и многие другие.

1.3 Описание данных, которые будут извлекаться:

Цель веб-скрапера будет заключаться в извлечении разнообразной информации с веб-сайтов, в зависимости от конкретных потребностей проекта. Это могут быть, например, структурированные данные о продуктах с электронной коммерции (например, наименования товаров, цены, описания, рейтинги и отзывы), информация о курсах валют, новости, данные о книгах или научных публикациях, информация с государственных порталов и многое другое.

Как часть процесса анализа проекта, необходимо определить конкретные целевые веб-сайты или ресурсы, с которых будут извлекаться данные, а также структуру и формат этих данных на веб-страницах. Это поможет определить, какие инструменты и методы скрапинга будут наиболее эффективными для извлечения необходимой информации, и какие аспекты структурированных данных будут важны для дальнейшего анализа и использования.

При разработке веб-скрапера на основе библиотек Beautiful Soup и requests будет уделено внимание оптимизации процесса извлечения данных, обработке неструктурированной информации на веб-страницах и преобразованию ее в удобочитаемый формат для дальнейшей обработки и анализа.

В данном проекте мы будем рассматривать пример использования данной технологии на примере скрайпинга сайти https://coinmarketcap.com/ru/

С помощью скрапинга сайта https://coinmarketcap.com/ru/ можно извлечь различные данные, такие как:

1. Информация о криптовалютах: Название, символ, текущая цена, изменение цены за последний час, 24 часа и 7 дней, объем торгов, рыночная капитализация и другие связанные данные.

2. Рейтинги и статистика: Рейтинги криптовалют, данные о рыночной капитализации и объемах торгов, изменения цен в процентном отношении за различные промежутки времени.

3. Данные о торговле: Информация о различных биржах, объемах торгов, количестве доступных криптовалютных пар и т.д.

4. Данные о криптовалютных компаниях: Сведения о компаниях, связанных с криптовалютами, новости, аналитика и пресс-релизы.

5. Информация о стабильных монетах: Данные по стабильным монетам, таким как Tether (USDT), US Dollar Coin (USDC) и другим.

Эти данные могут быть важны для инвесторов, трейдеров, исследователей и других участников рынка криптовалют для принятия информированных решений.

Кроме того, стоит отметить, что регулярное извлечение этих данных может помочь в анализе и прогнозировании динамики рынка криптовалют, а также предоставлять информацию для различных прикладных сфер, связанных с криптовалютами.

Эти данные могут быть упорядочены по различным критериям:

Данные с сайта CoinMarketCap могут быть упорядочены по различным критериям, включая:

1. Цена: Криптовалюты могут быть упорядочены по текущей цене или изменению цены за определенный период времени.

2. Рыночная капитализация: Криптовалюты можно отсортировать по их рыночной капитализации, что отражает общую стоимость монет в обращении.

3. Объем торгов: Данные могут быть упорядочены по объему суточных торгов, чтобы выявить самые активно торгуемые криптовалюты.

4. Изменение цены: Можно отсортировать криптовалюты по изменению цены за последний час, 24 часа, 7 дней и т.д.

5. Рейтинг: Криптовалюты также могут быть упорядочены по рейтингу, который учитывает различные факторы, такие как ликвидность, стабильность и другие.

Упорядочивание данных по этим критериям позволяет пользователям анализировать и сравнивать различные криптовалюты и принимать обоснованные инвестиционные решения.

При скрапинге информацию можно извлечь в различных форматах, включая:

1. Текстовый формат (TXT): Одним из наиболее распространенных способов извлечения информации при скрапинге является сохранение текстовых данных в формате .txt, который обеспечивает простой доступ к извлеченной информации.

2. Табличный формат (CSV, XLSX): Данные могут быть извлечены и сохранены в виде таблицы, используя форматы CSV (значения, разделенные запятыми) или XLSX (файлы Excel). Это особенно полезно для структурированных данных, таких как список товаров, каталогов и т.д.

3. Структурированный формат (JSON, XML): Извлеченные данные могут быть сохранены в структурированных форматах, таких как JSON или XML, что обеспечивает более сложную структурированную организацию информации и может быть полезно для обмена данными между различными системами.

4. Изображения и медиафайлы: В зависимости от конкретных потребностей скрапинга, также возможно извлечение изображений, аудио- или видеофайлов.

В контексте скрапинга, ускорение извлечения и обработки данных зависит от нескольких факторов, включая тип данных, объем информации, доступность API или структурированность веб-сайта. Однако, если рассматривать общий подход то TXT-файлы обычно являются самыми быстрыми для извлечения и обработки, поскольку они представляют собой простой текст без структурированных данных или форматирования. CSV-файлы также обычно обрабатываются быстро, поскольку они представляют собой таблицы с данными, которые могут быть легко интерпретированы программами для обработки данных. JSON и XML файлы могут потребовать немного больше времени для обработки из-за их структурированного характера, но в то же время они позволяют более гибкую обработку сложных данных.

CSV (Comma-Separated Values) - это текстовый формат, используемый для хранения табличных данных. В файле CSV каждая строка представляет

отдельную запись, а значения разделены запятыми (или другим разделителем, таким как точка с запятой). CSV-файлы могут содержать структурированные данные, такие как таблицы с данными, списки и каталоги.

Для быстрого преобразования CSV в табличные данные существует несколько подходов:

1. Использование библиотек для обработки данных: Python и другие языки программирования предлагают мощные библиотеки для чтения и записи CSV-файлов. Например, в Python библиотеки pandas, csv или openpyxl позволяют быстро и эффективно работать с CSV-файлами, преобразуя их в табличные данные для дальнейшей обработки.

2. Использование программных приложений для работы с данными: Существует множество программ, таких как Microsoft Excel, Google Sheets и другие, которые позволяют импортировать CSV-файлы и преобразовывать их в табличный формат для удобной работы с данными.

В целом, CSV-файлы представляют собой универсальный формат для хранения табличных данных, и их преобразование в табличный формат для обработки обычно происходит достаточно быстро благодаря доступным инструментам и библиотекам.

В данном проекте мы будет использовать формат хранения и извлечения данных с сайта indeed.com CSV-файлы.

2. Обзор технологий и методов

Этапы и методы скрапинга веб-сайтов включают в себя следующие технологии и подходы:

1. Выбор языка программирования: Для скрапинга можно использовать различные языки, такие как Python, JavaScript, Ruby или PHP. Python часто используется благодаря богатому выбору библиотек для скрапинга, таким как Beautiful Soup, Scrapy, Requests.

2. Использование библиотек и фреймворков: Beautiful Soup и Scrapy предоставляют удобные средства для извлечения и обработки данных из HTML и XML. Puppeteer, Cheerio и request-promise в JavaScript также являются эффективными инструментами для скрапинга.

3. XPath и CSS-селекторы: Использование XPath и CSS-селекторов для точного выбора элементов на веб-странице и извлечения нужной информации.

4. API-интеграция: Если веб-сайт предоставляет API, использование API для получения данных может быть более эффективным вариантом, чем скрапинг.

5. Использование прокси и антиблокировочных инструментов: При скрапинге важно быть готовым к блокировке доступа или ограничениям на веб-сайте. Использование прокси-серверов и антиблокировочных инструментов может помочь избежать этих проблем.

6. Автоматизация и планирование задач: Использование инструментов для планирования и автоматизации задач скрапинга, таких как cron в Unix или планировщик задач в Windows, позволяет выполнять скрапинг в удобное время и регулярно обновлять данные.

7. Обработка капчи и рекапчи: В случае, если веб-сайт имеет систему защиты от ботов в виде капчи или рекапчи, потребуется использование соответствующих инструментов или услуг, таких как сервисы решения капчи.

2.1Обзор библиотеки Beautiful Soup

Библиотека Beautiful Soup – это мощный инструмент на языке Python, предназначенный для парсинга HTML и XML документов. Основная её задача – это упрощение процесса извлечения данных из веб-страниц, что она делает, создавая дерево синтаксического анализа из исходного кода страницы и обеспечивая интуитивные методы для навигации, поиска и модификации этого дерева.

Beautiful Soup поддерживает множество парсеров HTML, например, встроенный в стандартную библиотеку Python, а также ряд внешних, таких как lxml и html5lib, что предоставляет разработчикам гибкость в выборе наиболее подходящих инструментов для их задач.

Помимо способности обрабатывать некорректно сформированный HTML код, что делает её особенно полезной для работы с реальными данными в интернете, которые часто не идеальны, Beautiful Soup также выделяется простотой использования. Простой и удобный интерфейс позволяет быстро осуществлять сложные запросы, как, например, извлекать все ссылки на странице или собирать информацию из определенных тегов.

Beautiful Soup - это мощная библиотека для парсинга HTML и XML, которая предлагает несколько преимуществ:

1. Простота использования: Beautiful Soup предоставляет простой и интуитивно понятный способ навигации по структуре HTML и извлечения нужных данных. Это делает библиотеку идеальным выбором для начинающих разработчиков.

2. Гибкость: Благодаря своей гибкости, Beautiful Soup может работать с неправильно созданным HTML, автоматически исправляя ошибки разметки и позволяя разработчикам сосредоточиться на извлечении данных.

Beautiful Soup отлично справляется с неправильно созданным HTML благодаря своей гибкости в обработке данных. Даже если HTML имеет ошибки разметки, например, не закрытые теги или неправильную вложенность, Beautiful Soup способна автоматически интерпретировать структуру документа и предоставлять доступ к его содержимому для извлечения данных.

Кроме того, при работе с неправильно созданным HTML, Beautiful Soup пытается самостоятельно исправить ошибки, что облегчает процесс парсинга и извлечения нужной информации. Таким образом, разработчики могут сосредоточиться на извлечении данных, не теряя время на исправление ошибок HTML-разметки.

3. Поддержка различных парсеров: Beautiful Soup поддерживает несколько различных парсеров, включая встроенные парсеры Python (по умолчанию), а также сторонние парсеры, такие как lxml. Это расширяет возможности библиотеки и позволяет выбирать наиболее подходящий парсер для конкретной задачи.

Beautiful Soup поддерживает несколько парсеров для обработки HTML и XML. В частности, библиотека поддерживает встроенные парсеры Python, такие как html.parser, а также сторонние парсеры, такие как lxml и html5lib. Каждый из этих парсеров имеет свои особенности и преимущества, что обеспечивает гибкость и возможность выбора наиболее подходящего парсера в зависимости от конкретной задачи и требований.

4. Удобство работы с данными: Благодаря своей возможности удобной навигации по структуре документа Beautiful Soup упрощает извлечение данных из сложных HTML-страниц и XML-документов.

5. Широкое сообщество и ресурсы: Благодаря своей популярности, Beautiful Soup обладает обширной документацией, множеством примеров использования и сообществом разработчиков, что облегчает поддержку и поиск решений для задач парсинга.

Тем не менее, существуют также некоторые недостатки. Одним из основных ограничений Beautiful Soup является её относительно низкая скорость по сравнению с более быстрыми парсерами, такими как lxml. Это может стать критическим при обработке большого объема данных.

Beautiful Soup имеет низкую скорость по сравнению с некоторыми более "низкоуровневыми" библиотеками, такими как lxml, из-за своего универсального и простого подхода к парсингу HTML и XML. Beautiful Soup фокусируется на предоставлении удобного и понятного API для извлечения данных из веб-страниц, что может отразиться на скорости обработки данных.

Тем не менее, в большинстве случаев небольшая потеря скорости компенсируется удобством использования и гибкостью библиотеки. Кроме того, оптимизация алгоритмов и использование более производительных парсеров,

таких как lxml, может помочь в увеличении скорости обработки данных при использовании Beautiful Soup.

К тому же, сама по себе Beautiful Soup не умеет выполнять HTTP запросы для получения веб-страниц, что значит, что для полноценного веб-скрейпинга её необходимо использовать в связке с другими библиотеками, такими как requests.

Одно из главных преимуществ использования Beautiful Soup с библиотекой requests заключается в том, что requests позволяет легко загружать содержимое веб-страницы, а затем Beautiful Soup упрощает извлечение необходимой информации из полученного HTML-кода. Синергия между этими двумя библиотеками обеспечивает простой и эффективный способ получения и обработки данных из Интернета.

Для глубокого изучения возможностей и получения инструкций по использованию Beautiful Soup можно обратиться к следующим ресурсам:

- Официальная документация Beautiful Soup, доступная на сайте crummy.com.

- Различные учебные руководства и статьи на тематических сообществах, таких как Habr или Medium, где опытные разработчики делятся своим опытом работы с библиотекой.

- Примеры кода и практические упражнения, которые можно найти на образовательных платформах или проекты на GitHub.

Используя этот инструмент в составе веб-скрейпера, разработчики способны эффективно собирать данные с разнообразных интернет-ресурсов, что является критически важным для обработки информации, автоматического сбора данных и последующего анализа.

Библиотека Beautiful Soup предоставляет множество возможностей для скрапинга веб-сайтов:

1. Навигация по HTML и XML: Beautiful Soup позволяет удобно перемещаться по структуре веб-страницы, находить нужные элементы, извлекать данные и анализировать их содержимое.

2. Парсинг данных: Благодаря своей гибкости, библиотека позволяет эффективно извлекать нужные данные из HTML и XML, а также проводить их дальнейшую обработку.

3. Работа с CSS-селекторами: Beautiful Soup позволяет использовать CSS-селекторы для поиска элементов веб-страницы, что облегчает процесс извлечения конкретных данных.

4. Обработка ошибок разбора HTML: Библиотека позволяет обрабатывать и исправлять ошибки разбора HTML, что упрощает процесс скрапинга веб-страниц.

5. Интеграция с различными парсерами: Beautiful Soup совместима с различными парсерами (например, html.parser, lxml, xml), что позволяет выбирать наиболее подходящий парсер в зависимости от конкретных потребностей.

Благодаря этим возможностям, Beautiful Soup является мощным инструментом для проведения скрапинга веб-сайтов и извлечения нужной информации из HTML и XML документов.

2.2 Обзор библиотеки requests

Библиотека requests в языке программирования Python представляет собой удобный инструмент для отправки HTTP-запросов и работы с веб-ресурсами. Она обеспечивает простой, но мощный интерфейс для работы с HTTP, позволяя легко взаимодействовать с веб-серверами, отправлять запросы о получении необходимых данных.

Преимущества использования библиотеки requests:

1. Простота использования: requests предоставляет лаконичный и интуитивно понятный API, что делает процесс отправки HTTP-запросов и обработки ответов непревзойденно простым и удобным.

2. Универсальность: requests поддерживает все основные методы HTTP, такие как GET, POST, PUT, DELETE, и др., что позволяет взаимодействовать с веб-серверами любого типа и обмениваться нужной информацией.

3. Поддержка сеансов: библиотека requests обеспечивает возможность управления состоянием сеанса, что особенно важно при работе с авторизацией, cookies и другими аспектами, требующими сохранения состояния между запросами.

4. Интеграция с другими инструментами: requests легко интегрируется с другими библиотеками Python, такими как Beautiful Soup, для эффективной обработки полученных данных.

5. Расширенные возможности: библиотека requests поддерживает различные опции для отправки запросов, включая пользовательские заголовки, параметры, файлы и многое другое, что обеспечивает широкие возможности для работы с веб-ресурсами.

Недостатки использования библиотеки requests:

1. Отсутствие асинхронности: requests является синхронной библиотекой, что может быть недостатком в некоторых сценариях, требующих асинхронного обмена данными с веб-ресурсами.

2. Необходимость дополнительных библиотек для работы с некоторыми форматами данных: для обработки некоторых типов данных, таких как JSON, XML, HTML, может потребоваться использование дополнительных библиотек.

Библиотека requests представляет собой важный инструмент для взаимодействия с веб-ресурсами в Python и может быть важным компонентом в процессе работы с данными из сети Интернет.

Библиотека requests в Python предоставляет широкий набор возможностей для веб-скрапинга сайтов. Вот некоторые из них:

1. Возможность отправки HTTP запросов: Библиотека requests позволяет легко отправлять HTTP запросы к веб-сайтам, получать HTML страницы и другие ресурсы.

2. Удобная работа с параметрами запроса: requests обеспечивает простой способ передачи параметров запроса, таких как заголовки, параметры URL, cookies и т.д.

3. Обработка cookies и сессий: Благодаря requests можно удобно работать с cookies и управлять сессиями, что полезно при скрапинге защищенных веб-сайтов.

4. Поддержка различных методов запросов: Библиотека поддерживает различные HTTP методы, такие как GET, POST, PUT, DELETE, что делает возможным взаимодействие с веб-сайтами через их API.

5. Обработка ответов: Requests позволяет удобно обрабатывать ответы от сервера, включая текст, JSON, изображения и другие типы файлов.

Эти возможности делают библиотеку requests очень мощным инструментом для веб-скрапинга и взаимодействия с веб-сайтами в целом.

2.3 Обзор библиотеки pandas

Библиотека Pandas - это мощный инструмент для анализа данных в языке программирования Python. Вот подробный обзор преимуществ и недостатков Pandas:

Преимущества Pandas:

1. Удобные структуры данных: Pandas предоставляет две основные структуры данных - Series и DataFrame, которые позволяют эффективно хранить и манипулировать табличными данными.

2. Мощные возможности анализа данных: Благодаря функциям для фильтрации, сортировки, группировки и преобразования данных, Pandas облегчает проведение различных аналитических операций.

3. Поддержка ввода/вывода данных: Библиотека Pandas позволяет читать данные из различных источников, таких как CSV, Excel, SQL, JSON, и другие, и экспортировать данные в различные форматы.

4. Обработка пропущенных значений: Pandas обеспечивает мощные инструменты для обработки пропущенных значений, включая методы для удаления, заполнения или интерполяции пропусков.

5. Широкие возможности визуализации: Библиотека Pandas интегрируется с другими инструментами для визуализации данных, такими как Matplotlib, что позволяет создавать информативные графики и диаграммы.

6. Интеграция с другими библиотеками: Pandas эффективно взаимодействует с другими популярными библиотеками Python, такими как NumPy, Scikit-learn, и другими инструментами для анализа данных и машинного обучения.

Недостатки Pandas:

1. Потребление памяти: При работе с большими объемами данных, Pandas может потреблять значительное количество памяти, что требует осторожного управления ресурсами.

2. Не всегда оптимальная производительность: Некоторые операции в Pandas могут быть менее эффективными по времени выполнения, особенно при обработке больших объемов данных, и требуют оптимизации кода.

3. Необходимость дополнительной работы с индексами: При работе с DataFrame требуется аккуратно обращаться с индексами, что может потребовать дополнительной работы при обработке и анализе данных.

Библиотека pandas предоставляет мощные возможности для обработки и анализа данных, что делает ее ценным инструментом для скрапинга сайтов. Вот несколько возможностей, которые открывает библиотека pandas:

1. Сбор и структурирование данных: pandas обеспечивает простые и удобные средства для сбора и структурирования данных, с помощью которых можно легко извлекать, фильтровать и преобразовывать данные после скрапинга.

2. Анализ и визуализация: библиотека pandas предоставляет широкие возможности для анализа собранных данных, включая статистические расчеты,

группировку, агрегацию и построение различных видов графиков для визуализации результатов.

3. Обработка пропущенных значений: pandas позволяет обрабатывать пропущенные или некорректные значения, заполнять их или удалять, что важно при анализе данных после скрапинга.

4. Интеграция с другими библиотеками: pandas хорошо интегрируется с другими библиотеками Python, такими как requests для получения данных из Интернета и beautifulsoup для парсинга веб-страниц.

Благодаря своей гибкости и функциональности, pandas помогает упростить процесс скрапинга, обработки и анализа данных с веб-сайтов.

2.4 Обзор библиотеки asyncio

Библиотека asyncio в Python представляет собой набор инструментов для написания асинхронного кода. Ее функционал включает в себя возможность создания корутин (асинхронных функций) с использованием ключевых слов async/await, организацию параллельного выполнения задач, управление событиями и ввод-выводом без блокирования основного потока выполнения. asyncio также предоставляет возможности для работы с сетевыми протоколами и работу с тайм-аутами.

Ключевыми функциональными возможностями библиотеки asyncio:

1. Асинхронные функции: Библиотека позволяет создавать асинхронные функции с использованием ключевого слова `async`, которые могут выполняться параллельно, ожидать другие асинхронные операции без блокирования процесса.

2. Корутины: asyncio поддерживает создание корутин (асинхронных генераторов) с помощью ключевого слова `async def`, обеспечивая гибкое выполнение асинхронных операций.

3. Event Loop: asyncio предоставляет цикл событий (event loop), который управляет выполнением асинхронных задач, позволяя им выполняться в неблокирующем режиме.

4. Ожидание событий: с помощью ключевого слова `await`, asyncio позволяет ожидать завершения асинхронных операций, таких как ввод-вывод, таймеры или другие асинхронные задачи.

5. Сетевые операции: asyncio предоставляет инструменты для работы с асинхронными сетевыми операциями, такими как соединение по протоколу TCP/UDP, выполнение HTTP-запросов, и другие.

6. Таймеры и планирование: библиотека позволяет планировать выполнение задач на определенное время или с интервалами, что полезно для периодических асинхронных операций.

В целом, asyncio обеспечивает основы для разработки асинхронных и многозадачных приложений в Python, позволяя эффективно использовать вычислительные ресурсы и обеспечивая отзывчивость приложений при выполнении сетевых операций или других I/O-интенсивных задач.

Эта библиотека используется для написания асинхронных приложений, таких как веб-серверы, сетевые приложения и другие приложения, требующие параллельной обработки задач. asyncio упрощает написание асинхронного кода, делая его более отзывчивым и эффективным по сравнению с традиционным синхронным кодом.

Библиотека asyncio в Python предоставляет инструменты для написания асинхронного кода. Ее основные преимущества:

1. Оперативность: asyncio позволяет выполнять асинхронные операции без блокирования основного потока выполнения, что повышает отзывчивость приложений.

2. Удобство: благодаря использованию корутин (async/await), asyncio упрощает написание асинхронного кода, делая его более читаемым и поддерживаемым.

3. Масштабируемость: asyncio предоставляет инструменты для организации параллельного выполнения задач, что позволяет эффективно использовать ресурсы системы.

Недостатки asyncio:

1. Сложность: из-за специфики работы с асинхронным кодом, начинающим разработчикам может быть сложно освоить asyncio и корутины.

2. Отсутствие поддержки некоторых библиотек: не все сторонние библиотеки имеют поддержку asyncio, что может потребовать поиска альтернативных решений.

Библиотека asyncio в Python открывает несколько возможностей для скрапинга сайтов:

1. Асинхронные запросы: asyncio позволяет выполнять асинхронные HTTP-запросы к веб-серверам, что может повысить эффективность и скорость скрапинга данных.

2. Параллельное выполнение: asyncio обеспечивает возможность параллельного выполнения нескольких скрапинг-запросов, что позволяет эффективно использовать ресурсы и сокращает общее время выполнения операций.

3. Управление вводом-выводом: благодаря механизму обработки ввода-вывода без блокировки, asyncio позволяет эффективно обрабатывать большое количество запросов к сайтам, минимизируя задержки.

4. Таймауты и задержки: библиотека предоставляет возможности установки таймаутов для запросов, а также планирования задержек между запросами, что полезно при скрапинге веб-сайтов для предотвращения блокировки IP-адреса.

5. Работа с парсерами: asyncio может быть использована с различными HTML-парсерами для извлечения данных из веб-страниц, что облегчает процесс скрапинга и анализа полученных данных.

Эти возможности позволяют разработчикам эффективно осуществлять скрапинг данных из веб-страниц с использованием asyncio, что может оказаться

особенно полезным при обработке больших объемов информации или при выполнении скрапинга в ресурсоемких сценариях.

2.5 Обзор библиотеки time

Библиотека time в Python представляет собой набор функций, которые позволяют работать с временем и измерять процессорное время выполнения программы. Она обеспечивает возможность работы с текущим временем, задержками и измерением интервалов времени, а также позволяет измерять процессорное время выполнения программы. Библиотека time является частью стандартной библиотеки Python и обладает простым в использовании интерфейсом.

Основные достоинства библиотеки time:

1. Простота использования: библиотека time содержит интуитивно понятные функции для работы с текущим временем, ожидания и измерения интервалов времени.

2. Переносимость: функционал библиотеки time доступен практически на всех платформах, поддерживаемых Python, что делает ее удобной для написания переносимого кода.

3. Измерение процессорного времени: библиотека time предоставляет возможность измерять процессорное время выполнения программы, что полезно для оптимизации производительности.

Недостатки библиотеки time:

1. Ограниченный функционал: для более сложных операций, таких как работа с датами и временем в различных часовых поясах, может потребоваться использование дополнительных библиотек, например, datetime.

Библиотека time в Python предоставляет возможности для управления временными задержками в процессе веб-скрапинга. Это позволяет лучше контролировать частоту запросов к веб-серверу, чтобы избежать перегрузки сервера и ограничений доступа, а также для организации регулярных

периодических обновлений данных. С помощью модуля time можно установить паузы между запросами к сайту, например, для соблюдения правил robots.txt, предотвращения блокировки IP-адреса или обеспечения безопасного и корректного скрапинга данных.

2.6 Обзор библиотеки logging

Библиотека logging в Python представляет собой инструмент для записи информационных сообщений, предупреждений, ошибок и отладочной информации в приложениях на Python. Она позволяет организовать систему логирования с различными уровнями важности сообщений. Библиотека logging обладает гибкими настройками форматирования вывода и возможностью направлять логи в различные цели, такие как файлы, консоль, сетевые службы и базы данных. Это делает ее мощным инструментом для отладки и мониторинга приложений.

Библиотека logging позволяет управлять логированием на уровне всего приложения, а также на уровне отдельных модулей. Она также поддерживает логирование в многопоточных и многопроцессорных приложениях.

Библиотека logging является частью стандартной библиотеки Python и широко используется для создания журналов действий и отслеживания работы приложений.

Основные достоинства библиотеки logging:

1. Гибкость настроек: библиотека позволяет гибко настраивать формат и уровни логирования для различных компонент приложения, что обеспечивает контроль над объемом и содержанием логов.

2. Поддержка различных целей вывода: logging предоставляет возможность направлять логи в файлы, консоль, сетевые службы, базы данных и другие цели, что делает ее удобной для различных сценариев использования.

3. Интеграция с многопоточностью и многопроцессорностью: библиотека позволяет безопасно использовать логирование в многопоточных и

многопроцессорных приложениях, предоставляя механизмы синхронизации доступа к логам.

Недостатки библиотеки logging:

1. Некоторая сложность настройки: для некоторых пользователей настройка библиотеки logging может показаться сложной из-за большого количества опций и возможностей.

2. Недостаточная документация: некоторые аспекты библиотеки могут не быть достаточно понятными из официальной документации.

Библиотека logging предоставляет мощные возможности для ведения журнала действий и ошибок в процессе веб-скрапинга. С её помощью можно осуществлять:

1. Запись информационных сообщений: Logging позволяет фиксировать различные информационные события, такие как начало или завершение операций, получение данных с веб-сайтов и другие ключевые шаги скрапинга.

2. Отслеживание ошибок и исключений: Библиотека позволяет регистрировать ошибки, предупреждения и исключения, что значительно облегчает процесс отладки и обнаружения проблем в коде скрапера.

3. Регулирование уровней сообщений: Logging поддерживает уровни сообщений (debug, info, warning, error, critical), позволяя гибко управлять записываемой информацией и при необходимости фильтровать сообщения.

4. Возможность логирования в файл: Библиотека позволяет записывать логи в файлы, что является удобным для дальнейшего анализа и контроля за процессом скрапинга.

В целом, библиотека logging обеспечивает удобный и надежный инструмент для отслеживания действий и проблем в процессе веб-скрапинга, что повышает уровень контроля и обеспечивает прозрачность операций.

2.7 Обзор библиотеки cryptography

Библиотека cryptography в Python представляет собой набор инструментов для реализации криптографических функций, таких как шифрование, генерация ключей, хэширование, подпись и проверка цифровых подписей. Она обеспечивает высокоуровневый интерфейс для выполнения различных криптографических операций, обеспечивая безопасное хранение и передачу данных. Библиотека cryptography также предоставляет реализации современных алгоритмов криптографии, обеспечивая высокий уровень безопасности.

Библиотека cryptography в Python предоставляет реализации различных алгоритмов криптографии, включая следующие:

1. Алгоритмы хэширования: SHA-2, SHA-3, BLAKE2, HMAC (имитовставки сообщения).

2. Симметричное шифрование: AES (Advanced Encryption Standard) в режимах ECB, CBC, CFB, OFB, CTR, GCM.

3. Асимметричное шифрование: RSA (генерация ключей, шифрование и подпись), Elliptic Curve Cryptography (ECC) с различными кривыми.

4. Цифровые подписи: подпись и верификация данных с использованием алгоритмов RSA, DSA, ECDSA.

5. Генерация безопасных случайных чисел (CSPRNG).

Библиотека cryptography ориентирована на обеспечение безопасной работы с криптографическими функциями и предоставляет высокоуровневый API для использования указанных алгоритмов.

Эта библиотека является надежным инструментом для обеспечения конфиденциальности и целостности данных, и широко используется в приложениях, работающих с криптографической защитой.

В библиотеке cryptography конфиденциальность и целостность данных обеспечиваются путем применения криптографических алгоритмов современных стандартов, а также соблюдения bewell-known best practices для обработки данных.

1. Шифрование данных: Для обеспечения конфиденциальности, используются симметричные и асимметричные алгоритмы шифрования, такие

как AES (Advanced Encryption Standard) для симметричного шифрования и RSA, ECDSA для асимметричного шифрования. Эти алгоритмы обеспечивают надежное шифрование данных, что делает их практически непригодными для несанкционированного доступа.

2. Хэширование данных: Для обеспечения целостности данных, библиотека использует криптографические хэширования такие как SHA-2 и SHA-3, которые создают уникальные хэш-суммы для данных, обеспечивая проверку их целостности.

3. Генерация безопасных случайных чисел: Для предотвращения предсказуемости ключей и других криптографических параметров, библиотека обеспечивает высококачественную генерацию безопасных случайных чисел (CSPRNG).

Помимо этого, для обеспечения повышенной безопасности, библиотека cryptography следует рекомендациям и стандартам криптографии, обновляется по мере появления новых методов атак и уязвимостей, и предоставляет средства для безопасного хранения криптографических ключей.

Библиотека cryptography в Python представляет собой мощный инструмент для выполнения различных криптографических операций. Ее основные достоинства:

1. Широкий спектр функциональности: библиотека cryptography предоставляет реализации современных алгоритмов криптографии, включая алгоритм AES, генерацию и проверку цифровых подписей, хэширование, генерацию ключей и многое другое.

2. Удобство использования: библиотека обладает понятным и лаконичным интерфейсом, что облегчает выполнение криптографических операций даже для относительно новых пользователей.

3. Отличная документация: официальная документация к библиотеке содержит обширное описание функций и примеров использования, что делает процесс изучения и работы с библиотекой более удобным.

Недостатки библиотеки cryptography:

1. Сложность определения настроек безопасности: для новичков может показаться сложным определение правильных настроек безопасности и выбор оптимальных алгоритмов для конкретных сценариев.

2. Требовательность к знаниям: из-за широкого спектра функциональности, библиотека cryptography может потребовать от пользователя глубоких знаний в области криптографии для эффективного и безопасного использования.

Библиотека cryptography сама по себе не предоставляет прямых возможностей для веб-скрапинга, так как ее основное назначение связано с криптографией и безопасностью данных. Однако, в контексте веб-скрапинга, библиотека cryptography могла бы быть использована для обеспечения безопасности данных, передаваемых или получаемых в процессе скрапинга.

Например, библиотека cryptography может быть использована для шифрования и защиты конфиденциальных данных, таких как аутентификационные данные, хранимые ключи или другие конфиденциальные сведения, используемые в процессе скрапинга. Такое применение помогает обезопасить передачу и хранение этих данных.

Таким образом, библиотека cryptography может играть роль в обеспечении безопасной передачи и хранения данных, полученных в результате веб-скрапинга, что особенно важно, если скрапинг используется для получения конфиденциальной информации.

2.8 Пояснение выбора данных технологий

Beautiful Soup была выбрана из-за своего удобного и интуитивно понятного API для работы с HTML и XML данными. Ее способность эффективно извлекать информацию из веб-страниц делает ее отличным выбором для веб-скрапинга. Преимущества Beautiful Soup включают простоту в поиске, навигации и модификации данных, таких как теги, классы, идентификаторы и текстовое содержимое на веб-страницах.

Когда речь идет о выполнении HTTP запросов, использование библиотеки requests является наиболее эффективным и удобным подходом в таких случаях, как отправка различных видов запросов (GET, POST, PUT и т. д.), управление заголовками и параметрами запросов, а также обработка ответов от веб-сервера. Requests предоставляет простой и понятный интерфейс для взаимодействия с веб-серверами, что делает его идеальным инструментом для выполнения HTTP запросов при веб-скрапинге.

Сочетание обеих библиотек, Beautiful Soup и requests, обеспечивает мощный инструментарий для разработки веб-скрапера. Beautiful Soup позволяет удобно извлекать данные из HTML и XML файлов, в то время как requests обеспечивает возможность отправки HTTP запросов и получения данных из сети. Эти библиотеки в совокупности позволяют эффективно извлекать и обрабатывать информацию из веб-страниц, делая их идеальным выбором для веб-скрапинга.

Использование библиотеки pandas для анализа полученных данных обеспечивает удобное средство для сохранения и обработки извлеченных данных. Данная библиотека позволяет легко структурировать данные в виде DataFrame, фильтровать и анализировать их с помощью встроенных методов, таких как:

1. Фильтрация данных:

- `query()`: Позволяет фильтровать DataFrame, используя выражения для отбора строк.

- `loc[]` и `iloc[]`: Предоставляют мощные средства для выбора и фильтрации данных по метке и позиции, соответственно.

- Методы сравнения (например, `equals()`, `isin()`, `between()`, `duplicated()`): Используются для выявления и фильтрации данных в соответствии со специальными условиями.

2. Анализ данных:

- Статистические методы: Например, `mean()`, `median()`, `std()`, `max()`, `min()`, `sum()` позволяют вычислять основные статистические показатели.

- Методы для работы с пропущенными значениями: `dropna()`, `fillna()` используются для удаления или заполнения пропущенных значений.

- Методы группировки: `groupby()`, `agg()` позволяют группировать данные и проводить агрегацию по определенным критериям.

3. Преобразование данных:

- `apply()`: Позволяет применять пользовательские функции к данным.

- Методы для работы с датами и временем: Pandas предоставляет множество методов для работы с датами и временем, такие как `to_datetime()`, `dt` accessors и другие.

Эти методы обеспечивают широкие возможности для анализа и фильтрации данных в pandas, делая библиотеку мощным инструментом для работы с табличными данными.

Библиотека time используется для управления временными задержками во время веб-скрапинга, в конкретном случае для использования временных задержек обращения к сайту.

Основные методы для работы с библиотекой time в Python:

1. time.time(): Возвращает текущее время в виде количества секунд, прошедших с начала эпохи (обычно 1 января 1970 года).

2. time.sleep(secs): Приостанавливает выполнение программы на заданное количество секунд, переданных в качестве аргумента.

3. time.localtime([secs]): Преобразует количество секунд, прошедших с начала эпохи, в структурированное представление времени в локальном временном поясе.

4. time.strftime(format[, t]): Преобразует структурированное представление времени (или текущее время, если t не указано) в строку согласно указанному формату.

5. time.gmtime([secs]): Преобразует количество секунд, прошедших с начала эпохи, в структурированное представление времени в UTC (всемирное координированное время).

Для обеспечения безопасности передачи и хранения извлеченных данных использована библиотека cryptography.

Основные методы для библиотеки cryptography в Python включают в себя:

1. Генерация ключей:

- cryptography.hazmat.primitives.asymmetric.utils.generate_elliptic_curve_private_key()

- cryptography.hazmat.backends.generate_rsa_private_key()

2. Шифрование и дешифрование:

- cryptography.fernet.Fernet.encrypt()

- cryptography.fernet.Fernet.decrypt()

3. Хеширование:

- cryptography.hazmat.primitives.hashes.HashAlgorithm.update()

- cryptography.hazmat.primitives.hashes.HashAlgorithm.finalize()

4. Генерация случайных данных:

- cryptography.hazmat.prng.SystemRandom()

5. Цифровые подписи:

- cryptography.hazmat.primitives.asymmetric.padding.PKCS1v15()

- cryptography.hazmat.primitives.asymmetric.utils.decode_dss_signature()

Библиотека логирования полезна для отслеживания работы веб-скрапера, сохранения журналов его действий, а также для решения проблем и отладки при необходимости.

Некоторые основные методы для библиотеки logging в Python включают в себя:

1. logging.debug(msg, *args, **kwargs): Метод для записи отладочных сообщений.

2. logging.info(msg, *args, **kwargs): Метод для записи информационных сообщений.

3. logging.warning(msg, *args, **kwargs): Метод для записи предупреждающих сообщений.

4. logging.error(msg, *args, **kwargs): Метод для записи сообщений об ошибках.

5. logging.critical(msg, *args, **kwargs): Метод для записи критических сообщений.

В случае, если требуется параллельное выполнение запросов к нескольким веб-сайтам, asyncio может быть использована для реализации асинхронного веб-скрапинга, что позволяет эффективно управлять множеством запросов без блокировки.

Некоторые из основных методов для библиотеки asyncio в Python включают в себя:

1. asyncio.get_event_loop(): Возвращает цикл событий, представляющий основной цикл событий.

2. loop.create_task(coro): Создает задачу, представляющую корутину coro и запускает ее асинхронное выполнение.

3. asyncio.sleep(delay, result=None, *, loop=None): Асинхронная функция, которая приостанавливает выполнение корутины на указанный интервал времени.

4. asyncio.wait_for(fut, timeout, loop=None): Ждет завершения Future, либо пока не истечет время ожидания.

5. asyncio.gather(*aws, loop=None, return_exceptions=False): Создает задачу, которая ожидает выполнения всех переданных корутин.

3. Описание проекта

Дипломная работа "Разработка веб-скрапера для извлечения данных с веб-сайтов с использованием библиотек Beautiful Soup, requests и pandas" имеет целью создание инструмента для автоматизированного сбора информации о криптовалютах с веб-сайта CoinMarketCap. В работе будет использоваться Python, а также библиотеки Beautiful Soup для парсинга HTML, requests для выполнения HTTP-запросов и pandas для обработки и анализа данных.

Дополнительно в рамках данной проектной работы были задействованы такие библиотеки как time, cryptography, logging, asyncio.

3.1 Постановка задачи

Цель проекта: "Разработка веб-скрапера для извлечения данных с веб-сайта CoinMarketCap с использованием библиотек Beautiful Soup, requests и pandas".

Поэтапные шаги реализации данного проекта:

1. Исследование: Проведите исследование структуры веб-сайта CoinMarketCap, который будет скрапиться, чтобы определить необходимые данные и способы их извлечения.

2. Установка библиотек: Установите все необходимые библиотеки, такие как Beautiful Soup, requests, pandas, time, cryptography, logging и asyncio, используя pip или другой менеджер пакетов.

3. Написание кода скрапера: Напишите скрипт на Python, который использовал бы библиотеки requests для загрузки HTML-страниц от CoinMarketCap, а затем применял бы Beautiful Soup для извлечения нужных данных о криптовалютах.

4. Внедрение безопасности (при необходимости): Если требуется защищенная передача данных или хранение данных, интегрируйте библиотеку cryptography для шифрования и безопасной передачи собранных данных.

5. Добавление журналирования: Внедрите модуль logging для учета работы вашего веб-скрапера, сохранения журналов и выявления проблем в процессе его работы.

6. Управление временными задержками: Используйте библиотеку time для добавления временных задержек между HTTP-запросами к веб-сайту CoinMarketCap, чтобы избежать блокировок и облегчить нагрузку на сервер.

7. Реализация асинхронности (при необходимости): В случае необходимости выполнения нескольких запросов параллельно, внедрите библиотеку asyncio для реализации асинхронного веб-скрапинга.

8. Тестирование и отладка: Проведите тщательное тестирование разработанного скрапера, убедившись в корректности извлечения и обработки данных, а также отсутствии ошибок и исключений.

9. Документирование: Документируйте код и процесс работы скрапера для облегчения его будущего использования и поддержки.

3.2 Архитектура проекта

Архитектура проекта для разработки веб-скрапера для извлечения данных о стоимости 100 криптовалют с веб-сайта CoinMarketCap с использованием библиотек Beautiful Soup и requests должна включать следующие ключевые компоненты:

а) Модуль отправки HTTP запросов:

б) Этот модуль будет использовать библиотеку requests для отправки GET запросов на веб-сайт CoinMarketCap с целью получения HTML страниц, содержащих информацию о 100 криптовалютах. Данные запросы могут быть выполнены с периодичностью для обновления информации.

в) Модуль парсинга HTML:

г) В этом модуле будет использоваться библиотека Beautiful Soup для анализа полученных HTML страниц и извлечения данных о стоимости каждой криптовалюты, а также других характеристик, таких как изменение цены за различные временные периоды, рыночная капитализация, объем торгов и другие показатели.

д) Хранилище данных:

е) Разработка механизма для хранения извлеченных данных в удобном формате, таком как CSV файл или база данных, позволит сохранить информацию для последующего использования и анализа.

ж) Автоматизация обновления данных:

з) Реализация автоматического механизма обновления данных с учетом периодичности и актуальности информации, чтобы обеспечить постоянную актуальность данных о стоимости криптовалют.

и) Тестирование и обработка ошибок:

к) Разработка механизмов тестирования веб-скрапера на различных обновлениях данных на веб-сайте CoinMarketCap, а также обработка возможных ошибок и изменений в структуре данных на веб-сайте.

л) Оптимизация производительности:

м) Оптимизация работы веб-скрапера для обеспечения эффективности и минимизации времени сбора и обработки данных, а также обработка возможных отправных ошибок или изменений структуры данных на веб-сайте CoinMarketCap.

н) Шифрование данных:

Внедрение модуля шифрования для обеспечения безопасности сохранения и передачи извлеченных данных о криптовалютах.

о) Журналирование действий:

Использование модуля logging для регистрации действий веб-скрапера, сохранения журналов и отслеживания работы системы.

п) Реализация асинхронности:

Внедрение asyncio для асинхронного выполнения запросов и обработки данных, что повысит производительность и эффективность веб-скрапера.

3.3 Описание работы скрапера

Разработка веб-скрапера для извлечения данных с веб-сайта CoinMarketCap

1. Настройка логирования: В данном блоке кода проводится настройка параметров и уровня логирования. Устанавливается файл, в который будут записываться журнал событий, уровень логирования и формат записи.

2. Генерация ключа шифрования: Для обеспечения безопасности данных происходит генерация ключа шифрования с использованием библиотеки Fernet.

3. Шифрование и расшифрование текста: Здесь представлены функции для шифрования и расшифрования текста с использованием сгенерированного ключа.

4. Отправка HTTP-запросов и обновление данных о криптовалютах: В блоке `update_crypto_data` отправляется HTTP-запрос на сайт CoinMarketCap, извлекается HTML-код страницы, затем с помощью библиотеки Beautiful Soup извлекаются данные о криптовалютах из таблицы. Заголовки и строки таблицы собираются в DataFrame, затем данные шифруются и сохраняются в файл.

5. Расшифрование и сохранение в формате CSV: Здесь представлена функция для расшифровки файла с данными о криптовалютах и их сохранения в формате CSV.

6. Обновление данных и ведение журнала: Цикл обновления данных работает бесконечно (`while True`), вызывая асинхронную функцию `update_crypto_data`, после чего происходит расшифровка и сохранение данных в файле CSV. В случае возникновения ошибки происходит логирование события.

7. Тестирование: набор тестов, который проверяет основной функционал веб-скрапера и его связанных операций.

4. Инструкция по применению

4.1 Установка необходимых библиотек

Для разработки веб-скрапера для извлечения данных с веб-сайта CoinMarketCap и сбора актуальной информации о стоимости 100 криптовалют с использованием библиотек Beautiful Soup и requests, необходимо выполнить следующие шаги:

1. Установка Python:

Убедитесь, что на вашем компьютере установлен Python. Если нет, скачайте и установите Python с его официального сайта: https://www.python.org/downloads/.

2. Установка библиотек requests, Beautiful Soup, pandas, cryptography:

Откройте командную строку и выполните следующие команды для установки библиотек requests и Beautiful Soup:

```
pip install requests
pip install beautifulsoup4
pip install pandas
pip install cryptography
```

3. Импорт библиотек:

В вашем Python скрипте импортируйте библиотеки requests, Beautiful Soup, pandas, cryptography, time, logging, asyncio:

```
import requests
import asyncio
from bs4 import BeautifulSoup
import pandas as pd
import time
```

```
import logging
from cryptography.fernet import Fernet
```

4. Отправка HTTP-запроса:

Используйте библиотеку requests для отправки GET-запроса на веб-сайт CoinMarketCap для получения HTML-страницы с данными о криптовалютах:

```
url = 'https://coinmarketcap.com/'
response = requests.get(url)
```

5. Парсинг HTML:

Примените библиотеку Beautiful Soup для анализа полученной HTML-страницы и извлечения информации о стоимости, изменении цены за различные временные периоды, рыночной капитализации и других характеристиках для каждой из 100 криптовалют.

```
soup = BeautifulSoup(response.text, 'html.parser')
```

6. Хранение данных:

Разработайте механизм для сохранения извлеченных данных в удобном формате, например, в файле CSV или базе данных, для дальнейшего анализа и использования.

7. Тестирование

Для данного проекта разработаны следующие типы тестов:

1. `test_website_availability`: Данный тест проверяет доступность веб-сайта CoinMarketCap по заданному URL с использованием библиотеки requests. Он выполняет проверку статус-кода ответа. В случае, если статус-код не равен 200, тест выдаст сообщение об ошибке.

2. `test_table_existence`: Этот тест проверяет наличие таблицы на веб-странице CoinMarketCap по заданному URL. Сначала он делает GET-запрос для

получения HTML-кода страницы, затем с помощью BeautifulSoup ищет таблицу. Если таблица не найдена, тест генерирует сообщение об ошибке.

3. `test_csv_creation`: Данный тест проверяет создание CSV-файла на основе данных, полученных из таблицы на веб-странице. Он также отправляет GET-запрос для получения HTML-кода, извлекает данные из таблицы с помощью BeautifulSoup, создает DataFrame с помощью pandas, и сохраняет его в CSV-файл. Если файл успешно создан, тест проходит. В противном случае, он выдает сообщение об ошибке.

4. `test_encryption`: Этот тест проверяет функционал шифрования и дешифрования с использованием библиотеки cryptography. Он генерирует ключ шифрования, шифрует текст, а затем расшифровывает его. Если шифрование и дешифрование прошли успешно, тест считается пройденным.

Каждый тест написан для проверки определенного аспекта функциональности приложения, таких как доступность веб-сайта, наличие таблицы на странице, создание CSV-файла с данными и корректность работы шифрования и дешифрования данных.

8. Автоматическое обновление данных:

В данном коде автоматическое обновление данных осуществляется путем постоянного выполнения цикла while. Внутри цикла while вызывается асинхронная функция "update_crypto_data()", которая отправляет запрос на веб-сайт "https://coinmarketcap.com/ru/", получает HTML-код, извлекает данные о криптовалютах, создает DataFrame с помощью библиотеки pandas, шифрует данные и сохраняет их в файл "crypto_data.encrypted".

После этого вызывается функция "decrypt_and_save_csv()", которая расшифровывает файл "crypto_data.encrypted", сохраняет его в формате CSV в файл "decrypted_crypto_data.csv".

После выполнения этих операций программа ожидает 3600 секунд (1 час) с помощью функции time.sleep(3600), прежде чем повторить процесс обновления данных.

Таким образом, данный код создает автоматическое обновление данных о криптовалютах с интервалом в 1 час, используя асинхронную обработку для эффективного управления операциями получения данных из сети.

4.2 Пример кода и пояснение

См. Папку Code

4.3 Инструкции по запуску и использованию скрапера

Для запуска и использования данного веб-скрапера, выполните следующие шаги:

1. Установка необходимых библиотек: убедитесь, что у вас установлены библиотеки requests, beautifulsoup4, pandas и cryptography. Вы можете установить их с помощью pip, используя следующие команды:

```
pip install requests
pip install beautifulsoup4
pip install pandas
pip install cryptography
```

2. Запустите скрипт: после установки необходимых библиотек, сохраните предоставленный код в файле с расширением ".py". Затем запустите скрипт, используя интерпретатор Python.

3. Мониторинг журнала: в процессе выполнения скрипта журнал (scraper_log.log) будет вести записи о процессе обновления данных о криптовалютах, включая информацию об успешных операциях или ошибках.

4. Разрыв выполнения: скрипт будет выполнять обновление данных и ведение журнала каждый час. Вы можете прервать выполнение скрипта, нажав на клавиатуре "Ctrl+C".

Эти шаги позволят вам успешно запустить и использовать скрапер для обновления данных о криптовалютах с интервалом в 1 час и вести журнал всех операций.

5. План разработки проекта

1. Определение требований:

- Источник данных: веб-сайт 'https://coinmarketcap.com/ru/'.

- Формат и структура данных: данные о криптовалютах извлекаются из таблицы на веб-странице.

- Частота обновления данных: обновление данных каждый час.

2. Разработка функций скрапера:

- Функция для запроса данных: использование библиотеки `requests` и асинхронного запроса `asyncio` для получения HTML-страницы.

- Функция для анализа и извлечения данных: использование `BeautifulSoup` для поиска и извлечения данных из HTML-кода.

- Функция для сохранения данных: создание DataFrame с помощью `pandas` и шифрование/расшифрование данных с использованием `cryptography.fernet`.

3. Реализация автоматического обновления:

- Использование цикла `while` и `time.sleep` для периодического вызова функции обновления данных.

- Обработка и регистрация возможных ошибок с использованием `logging`.

4. Тестирование и отладка:

- Тестирование скрапера для проверки корректности выполнения каждого шага.

- Проверка точности и актуальности данных, получаемых скрапером.

5. Документация и использование:

- Подробная инструкция по установке необходимых библиотек, запуску скрипта и интерпретации журнала логирования.

- Возможность публикации проекта в репозитории для доступа и совместной работы.

Данный план учитывает основные этапы разработки скрапера, начиная от определения требований и разработки функционала до тестирования, отладки и конечного использования.

6. Заключение

Представленный веб-скрапер является мощным инструментом для автоматического сбора и обновления данных о криптовалютах с веб-сайта CoinMarketCap. Он обладает следующими ключевыми функциями:

1. Автоматическое обновление данных: Скрапер регулярно обновляет информацию о криптовалютах и сохраняет ее в зашифрованном формате.

2. Шифрование данных: Перед сохранением в файл данные шифруются с использованием библиотеки cryptography.fernet, обеспечивая безопасность и защиту конфиденциальности данных.

3. Логирование: Ведется запись всех операций скрапера в файл журнала, что позволяет отслеживать успешные операции и ошибки.

4. Тестирование: Представлены unit-тесты для проверки доступности веб-сайта, наличия таблицы на странице, создания CSV-файла и правильности шифрования данных.

Для дальнейшего совершенствования проекта возможно внедрение дополнительных функций, таких как мониторинг изменений на веб-сайте и автоматическое оповещение об аномалиях, использование асинхронных запросов для повышения эффективности скрапинга, а также дальнейшее тестирование для обеспечения стабильной работы скрапера в различных сценариях использования.

6.1 Сводка целей проекта

Сводка целей данного проекта:

1. Скрапинг данных о криптовалютах: Основной целью проекта является регулярный сбор данных о криптовалютах с веб-сайта CoinMarketCap. Эти данные будут использоваться для отслеживания и анализа текущего состояния рынка криптовалют.

2. Шифрование и безопасное хранение данных: После скрапинга данных о криптовалютах они шифруются и сохраняются в зашифрованном формате. Это помогает обеспечить безопасность и конфиденциальность собранных данных.

3. Журналирование и обработка ошибок: Цель ведения логов состоит в отслеживании всех операций скрапинга и обнаружении возможных ошибок или проблем, чтобы обеспечить надежную и стабильную работу процесса сбора данных.

4. Тестирование и поддержание качества: Проект включает в себя набор тестов, которые проверяют различные аспекты функциональности скрапера, включая доступность веб-сайта, наличие таблицы на странице, создание CSV-файла, а также корректность шифрования данных. Тесты помогают поддерживать высокое качество и надежность проекта.

5. Автоматизация обновления данных: Проект предусматривает автоматическое обновление данных о криптовалютах каждый час, чтобы обеспечить актуальность информации.

Цели проекта объединяют усилия по сбору, защите, и обработке данных о криптовалютах для обеспечения их надежности и актуальности.

6.2 Краткое изложение результатов

Этот код выполняет несколько задач:

1. Процесс скрапинга: Асинхронный процесс обновления данных о криптовалютах с веб-сайта CoinMarketCap с использованием библиотеки requests для выполнения HTTP-запросов и BeautifulSoup для извлечения информации из HTML-страниц.

2. Шифрование данных: Данные о криптовалютах шифруются с использованием библиотеки cryptography и ключа шифрования Fernet, далее сохраняются в зашифрованном формате.

3. Логирование: Отслеживание операций и обработка ошибок осуществляется через журналирование, где записываются события и возможные проблемы.

4. Обновление данных: Автоматическая периодическая обработка данных каждый час.

5. Юнит-тестирование: Набор тестов для проверки различных аспектов функциональности скрапера, включая доступность веб-сайта, наличие таблицы на странице, создание CSV-файла и правильность шифрования.

Эти шаги объединяются для создания автоматизированного процесса сбора, защиты и обновления данных о криптовалютах.

7. Источники

- Subirana, B. (2015). Operating on a higher ethical level: Codes of ethics can provide real-world guidance in data mining. Marketing News, 49(11), 24-24.
- Provost, F., & Fawcett, T. (2013). Data science for business: what you need to know about data mining and data-analytic thinking. O'Reilly Media, Inc.
- Pasquier, M., Axelsson, S., & Snekkenes, E. (2017). Assessing data privacy in emerging applications using data mining.
- Pomerantz, J., & Marchionini, G. (2007). The digital library as place. Journal of Documentation, 63(4), 505-533.
- Tufekci, Z. (2014). Big questions for social media big data: Representativeness, validity and other methodological pitfalls. ICWSM, 14(13-14), 505-514.
- Официальная документация Python по работе с CSV: Официальная документация Python предоставляет обширную

информацию о библиотеках для работы с CSV, таких как pandas, csv и openpyxl, а также подробные примеры использования.

- Документация программных приложений: Многие программы для работы с данными, такие как Microsoft Excel и Google Sheets, предоставляют свои собственные руководства и статьи о работе с CSV-файлами.
- Использование официальной документации Python и учебных материалов по обработке данных обеспечит надежные и проверенные источники информации о работе с CSV-файлами.
- Официальная документация: Посетите официальный сайт Beautiful Soup (crummy.com) для доступа к официальной документации, примерам кода и рекомендациям по использованию этой библиотеки.
- Учебные руководства и статьи: Поиск учебных материалов на тематических ресурсах, таких как Habr или Medium, где опытные разработчики делятся советами, практическими примерами и бест-практис по использованию Beautiful Soup.
- Официальная документация: посетите официальный сайт requests (docs.python-requests.org), чтобы получить доступ к подробной документации, примерам кода и руководствам.
- Проекты на GitHub и примеры кода: изучите открытые проекты, содержащие примеры использования библиотеки requests, на платформе GitHub и в различных источниках.
- Официальная документация Pandas: https://pandas.pydata.org/pandas-docs/stable/
- "Python for Data Analysis" by Wes McKinney, O'Reilly Media, 2017.
- "Effective Python: 90 Specific Ways to Write Better Python" by Brett Slatkin - книга содержит ценные советы по использованию asyncio.

- "Real Python" - онлайн-ресурс, содержащий статьи и руководства по asyncio.
- "Python 3 Documentation" - официальная документация Python.
- "A Whirlwind Tour of Python" by Jake VanderPlas - книга содержит обзор основных библиотек Python, включая time.
- "Logging in Python" by Vinay Sajip - книга содержит подробное описание работы с библиотекой logging
- "Cryptography Official Documentation" - официальная документация библиотеки cryptography.
- "Cryptography Official Documentation" - официальная документация библиотеки cryptography.