1Время выполнения: 0.0024609565734863 28 секу
2Время выполнения: 0.0010318756103515625 секунд
3Время выполнения: 0.024649143218994 14 секунд

Осн. Время выполнения: 0.0010318756103515625 секунд

Иерархическая структура данных — это способ организации данных, в котором элементы располагаются друг относительно друга в виде дерева, представляющего родительские и дочерние отношения. В таком представлении каждый элемент может иметь один или несколько дочерних элементов, иерархически подчинённых родительскому элементу.
(Парсер возвращает результат в виде структуры данных Python, которую можно легко обрабатывать. Полученный результат является иерархической структурой (например, словарь для объекта или список для массива), что позволяет далее использовать его для трансформации в другой формат, такой как XML.)

1) Markup - это общий термин, обозначающий системы для разметки текста, используют теги, чтобы структурировать документ. Markdown - это облегченный язык разметки, разработанный для быстрого форматирования текста, обычно без сложных тегов, что делает его более читаемым в исходной форме.

2) Протокол буферов (PROTOBUF) представляет данные в бинарном формате, что делает его более компактным и более быстрым в сравнении с текстовыми форматами. Он также обеспечивает строгую схему, что позволяет лучше управлять версиями и изменениями в структуре данных. (подробно в sm)

3) CSV (Comma-Separated Values) использует запятые в качестве разделителей между значениями, тогда как TSV (Tab-Separated Values) использует табуляцию. Это влияет на то, как данные обрабатываются, и какие символы могут быть допустимы внутри значений.

4) Новые форматы представления данных возникают из-за изменений в технологии, требованиям безопасности, эффективности обработки данных, потребностям пользователей и интеграции с современными системами. Новые форматы могут предлагать лучшие возможности для оптимизации, компрессии, универсальности и поддержки.

5) В XML символы ‘>’ и ‘<’ представляются как `&gt;` и `&lt;` соответственно. Это необходимо для корректного отображения этих символов, чтобы они не интерпретировались как часть разметки.

6) Сериализация данных - это процесс преобразования структуры данных или объекта в формат, который может быть сохранен на диске или передан по сети, и позже десериализован обратно в оригинальный объект. Это позволяет обмениваться данными между различными системами и хранить их для последующего использования.

7) В YAML комментарии обозначаются знаком решетки (#). Все, что следует за этим знаком до конца строки, считается комментарием и игнорируется при обработке.

8) В Markdown заголовки создаются с помощью символа `#`, количество решеток соответствует уровню заголовка (от 1 до 6). Для оформления кода используется обратная кавычка (или тройные обратные кавычки для многострочного кода). Полужирный текст обрамляется двойными звездочками (*) или двойными подчеркиваниями (__), курсивный текст - одинарными звездочками (*) или подчеркиваниями (_), а зачеркнутый текст - тильдой (~) с обеих сторон.

9) В современных мессенджерах, таких как Viber, WhatsApp и Telegram, используются форматы обмена данными, такие как JSON для сообщений, а также бинарные форматы для медиафайлов, такие как изображения и видео.(подробно в sm)

10) Аббревиатура SVG расшифровывается как Scalable Vector Graphics, что переводится как "векторная графика, масштабируемая без потери качества".

11) Пример использования тега для создания гиперссылки в HTML: `<a href="https://example.com">Это гиперссылка</a>`.

12) JSON-текст может представлять собой различные структуры данных, включая объекты и массивы. В закодированном виде JSON может представлять как простые данные (строки, числа, логические значения), так и сложные структуры (вложенные объекты и массивы), что позволяет создавать иерархические структуры данных.

Пример — арифметические выражения [править | править код]

Рассмотрим простой язык, определяющий ограниченное подмножество арифметических формул, состоящих из натуральных чисел, скобок и знаков арифметических действий. Стоит заметить, что здесь в каждом правиле с левой стороны от стрелки → стоит только один нетерминальный символ. Такие грамматики называются контекстно-свободными.

Терминальный алфавит:

```
Σ = {'0','1','2','3','4','5','6','7','8','9','+','−','*','/','(',')'}
```

Нетерминальный алфавит:

```
{ ФОРМУЛА, ЗНАК, ЧИСЛО, ЦИФРА }
```

Правила:

```
1. ФОРМУЛА → ФОРМУЛА ЗНАК ФОРМУЛА               (формула есть две формулы, соединенные знаком)
2. ФОРМУЛА → ЧИСЛО                              (формула есть число)
3. ФОРМУЛА → ( ФОРМУЛА )                        (формула есть формула в скобках)
4. ЗНАК → + | − | * | /                         (знак есть плюс или минус, или умножить, или разделить)
5. ЧИСЛО → ЦИФРА                                (число есть цифра)
6. ЧИСЛО → ЧИСЛО ЦИФРА                          (число есть число и цифра)
7. ЦИФРА → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 (цифра есть 0 или 1, или ... 9 )
```

Начальный нетерминал:

```
ФОРМУЛА
```

Вывод:

Выведем формулу (12+5) с помощью перечисленных правил вывода. Для наглядности, стороны каждой замены показаны попарно, в каждой паре заменяемая часть подчеркнута.

ФОРМУЛА $\xrightarrow{3}$ (ФОРМУЛА)

(ФОРМУЛА) $\xrightarrow{1}$ (ФОРМУЛА ЗНАК ФОРМУЛА)

(ФОРМУЛА ЗНАК ФОРМУЛА) $\xrightarrow{4}$ (ФОРМУЛА + ФОРМУЛА)

(ФОРМУЛА + ФОРМУЛА) $\xrightarrow{2}$ (ФОРМУЛА + ЧИСЛО)

(ФОРМУЛА + ЧИСЛО) $\xrightarrow{5}$ (ФОРМУЛА + ЦИФРА)

(ФОРМУЛА + ЦИФРА) $\xrightarrow{7}$ (ФОРМУЛА + 5)

(ФОРМУЛА + 5) $\xrightarrow{2}$ (ЧИСЛО + 5)

(ЧИСЛО + 5) $\xrightarrow{6}$ (ЧИСЛО ЦИФРА + 5)

(ЧИСЛО ЦИФРА + 5) $\xrightarrow{5}$ (ЦИФРА ЦИФРА + 5)

(ЦИФРА ЦИФРА + 5) $\xrightarrow{7}$ (1 ЦИФРА + 5)

(1 ЦИФРА + 5) $\xrightarrow{7}$ (1 2 + 5)

Аналитические грамматики

Порождающие грамматики — не единственный вид грамматик, однако наиболее распространенный в приложениях к программированию. В отличие от порождающих грамматик, **аналитическая (распознающая) грамматика** задает алгоритм, позволяющий определить, принадлежит ли данное слово языку. Например, любой регулярный язык может быть распознан при помощи грамматики, задаваемой конечным автоматом, а любая контекстно-свободная грамматика — с помощью автомата со стековой памятью. Если слово принадлежит языку, то такой автомат строит его вывод в явном виде, что позволяет анализировать семантику этого слова.

https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0