Câu1 **XML** là một ngôn ngữ đánh dấu mở rộng, được sử dụng để tạo các tài liệu có cấu trúc. Nói một cách đơn giản, XML cho phép bạn tạo ra một ngôn ngữ đánh dấu riêng để mô tả dữ liệu của bạn theo cách mà cả con người và máy tính đều có thể hiểu được.

Câu 2 **DOM**) là một giao diện lập trình ứng dụng (API) được sử dụng để truy cập và thao tác với các tài liệu HTML và XML. Nói một cách đơn giản, DOM cung cấp một cách để các ngôn ngữ lập trình (như JavaScript) tương tác với các trang web và các tài liệu XML.

Câu 3**XML** là một ngôn ngữ đánh dấu mở rộng, được sử dụng để tạo các tài liệu có cấu trúc. Cấu trúc của một tài liệu XML khá đơn giản và tuân theo một số quy tắc cú pháp nghiêm ngặt

Câu 4 **JSON**, là một định dạng trao đổi dữ liệu dựa trên văn bản (text-based data format) được sử dụng rộng rãi trong lập trình web. JSON được thiết kế để dễ đọc và dễ hiểu cho cả con người và máy tính, đồng thời cũng rất dễ dàng để các ngôn ngữ lập trình khác nhau phân tích cú pháp.

Câu 5**Hạn chế của việc triển khai JSON trong Python**

Mặc dù JSON là một định dạng dữ liệu phổ biến và linh hoạt, việc sử dụng JSON trong Python vẫn tồn tại một số hạn chế nhất định. Dưới đây là một số điểm cần lưu ý:

**1. Khả năng biểu diễn kiểu dữ liệu:**

* **Giới hạn các kiểu dữ liệu cơ bản:** JSON chỉ hỗ trợ các kiểu dữ liệu cơ bản như số, chuỗi, boolean, null, mảng và đối tượng. Các kiểu dữ liệu phức tạp hơn như ngày tháng, thời gian, số thập phân chính xác cao hoặc các đối tượng tùy định thường cần được chuyển đổi sang các dạng tương đương trong JSON trước khi lưu trữ hoặc truyền đi.
* **Không hỗ trợ các kiểu dữ liệu tùy định:** JSON không có cơ chế trực tiếp để biểu diễn các lớp hoặc cấu trúc dữ liệu tùy định trong Python. Điều này có thể dẫn đến việc phải thực hiện thêm các bước mã hóa và giải mã khi làm việc với các đối tượng phức tạp.

**2. Hiệu suất:**

* **Phân tích cú pháp:** Mặc dù JSON có cú pháp đơn giản, nhưng việc phân tích cú pháp một chuỗi JSON lớn vẫn có thể tốn thời gian, đặc biệt là khi thực hiện nhiều lần.
* **Mã hóa và giải mã:** Quá trình chuyển đổi giữa các đối tượng Python và JSON cũng có thể ảnh hưởng đến hiệu suất, đặc biệt khi làm việc với các tập dữ liệu lớn.

**3. Bảo mật:**

* **Tiếp xúc trực tiếp:** Nếu JSON được sử dụng để truyền dữ liệu nhạy cảm, việc bảo mật dữ liệu trở nên quan trọng. Việc mã hóa dữ liệu trước khi truyền đi và xác thực dữ liệu sau khi nhận được là cần thiết để ngăn chặn các cuộc tấn công như injection.

**4. Xác thực:**

* **Thiếu cơ chế xác thực mạnh:** JSON không có cơ chế xác thực dữ liệu mạnh mẽ như XML Schema. Điều này có thể dẫn đến các lỗi khó phát hiện nếu dữ liệu JSON không đúng định dạng.

**5. Kích thước file:**

* **Dữ liệu lặp lại:** Nếu dữ liệu có nhiều phần tử lặp lại, JSON có thể tạo ra các file có kích thước lớn hơn so với các định dạng nén như Protocol Buffers hoặc MessagePack.

**Cách khắc phục:**

* **Sử dụng các thư viện hỗ trợ:** Các thư viện như orjson có thể cung cấp hiệu suất phân tích cú pháp tốt hơn so với thư viện json mặc định của Python.
* **Tối ưu hóa cấu trúc dữ liệu:** Bằng cách thiết kế cấu trúc dữ liệu JSON hợp lý, bạn có thể giảm thiểu kích thước file và cải thiện hiệu suất.
* **Sử dụng các định dạng dữ liệu khác:** Trong một số trường hợp, các định dạng dữ liệu khác như Protocol Buffers hoặc MessagePack có thể phù hợp hơn cho việc truyền dữ liệu hiệu quả và bảo mật.
* **Xác thực dữ liệu:** Thực hiện kiểm tra dữ liệu đầu vào để đảm bảo rằng dữ liệu JSON tuân thủ các yêu cầu về định dạng và cấu trúc.