XML

XML (eXtensible Markup Language) is another way of exchanging data between browsers and servers (JSON is an alternative to XML).

XML is a markup language like HTML, so it contains data, and information on how to structure that data, but not how it is displayed. Hence we need an API to extract data from an XML file.

They are hierarchical in structure.

The document usually contains a prolog tag containing meta data, such as version, character encoding and associated style sheet.

The next tag will be the root(`data` in this case) tag which will contain all other tags of the document.

Each tag is completely flexible in it's naming unlike HTML which has a pre-defined set of tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <employee name="Alpha">
        <email>alpha@aicore.com</email>
        <department>HR</department>
        <age>36</age>
    </employee>
    <employee name="Bravo">
        <email>bravo@aicore.com</email>
        <department>sales</department>
        <age>23</age>
    </employee>
    <employee name="Charlie">
        <email>charlie@aicore.com</email>
        <department>accounts</department>
        <age>44</age>
    </employee>
    <employee name="Delta">
        <email>delta@aicore.com</email>
        <department>reception</department>
        <age>51</age>
    </employee>
</data>
```

Document: The root tag opens the document in this case `<data>` and the ending tag `</data>` closes it.

Node: Each tag containing other tags is a node tag here `<employee>` is a node tag.

Elements: Elements such as `<email>alpha@aicore.com</email>` and `<age>36</ag` considered elements.

Content: The data between the elements tags are considered content. In the email element `<email>alpha@aicore.com</email>`, the string `alpha@aicore.com` is considered the content.

Pandas

```
df = parse_XML("employees.xml", ["name", "email", "department", "age"])
df = pd.read_xml("employees.xml", attrs_only=True, elems_only=True)
df.to_xml("employees_df_export")
```