



PYTHON

File Processing

XML - Creating and processing XML files

FILE PROCESSING

XML - Creating and processing XML files

FILE PROCESSING

- **XML:**

- XML (*Extensible Markup Language*) es un lenguaje de marcado diseñado para almacenar y transportar datos.
- Permite definir etiquetas propias que facilitan la lectura del documento. XML es un estándar recomendado por el W3C.

- **Estructura:**

- **Prólogo:** la primera línea (**opcional**) del documento. En el prólogo, se puede especificar la codificación de caracteres.
 - **Elemento raíz:** el documento XML debe tener **un elemento raíz** que contenga todos los demás elementos.
 - **Elementos:** estos consisten en etiquetas de apertura y cierre. Los elementos incluyen **texto**, **atributos** y otros **elementos secundarios** (otros elementos).
 - **Atributos:** se colocan en las etiquetas de apertura. Consisten en pares clave-valor.
- **NOTA:** Cada etiqueta XML de apertura debe tener su correspondiente etiqueta de cierre.

FILE PROCESSING

- XML. Ejemplo

```
<?xml version="1.0"?>
<data>
  <book title="The Little Prince">
    <author>Antoine de Saint-Exupéry</author>
    <year>1943</year>
  </book>
  <book title="Hamlet">
    <author>William Shakespeare</author>
    <year>1603</year>
  </book>
</data>
```

FILE PROCESSING

- La biblioteca estándar de Python ofrece varios módulos para el manejo de XML:
 - **xml.etree.ElementTree**. Dispone de un API muy sencillo para crear y analizar documentos XML. Representa el documento en una estructura de datos en forma de árbol.
 - **xml.dom.minidom**. Proporciona una implementación mínima del API DOM (*Document Object Model*). Representa el documento en una estructura de datos en forma de árbol.
 - **xml.sax**. Proporciona una implementación del API SAX (*Simple Api for XML*). Está basado en eventos.

FILE PROCESSING

- **xml.etree.ElementTree.**
 - Procesamiento y obtención del nodo raíz.
 - **Opción 1. Métodos `parse` y `getroot`.**

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()
```

- **Opción 2. Método `fromstring`**

```
import xml.etree.ElementTree as ET
```

```
root = ET.fromstring(your_xml_as_string)
```

FILE PROCESSING

- **xml.etree.ElementTree.**

- Accediendo al contenido:

- Un elemento es un iterable compuesto de elementos.
 - Los elementos pueden tener **atributos**
 - El atributo **tag** proporciona el nombre del elemento.
 - El atributo **attrib** proporciona un diccionario con los atributos.

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')
root = tree.getroot()
print('The root tag is:', root.tag)
print('The root has the following children:')
for child in root:
    print(child.tag, child.attrib)
```



```
The root tag is: data
The root has the following children:
book {'title': 'The Little Prince'}
book {'title': 'Hamlet'}
```

FILE PROCESSING

- **xml.etree.ElementTree.**

- Accediendo al contenido:

- El contenido textural de un elemento se obtiene con el atributo **text**.
 - Se puede acceder a los sub-elementos de un elemento a través del **índice** (se pueden indicar en varios niveles [0][1]).

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()  
print("My books:\n")  
for book in root:  
    print('Title: ', book.attrib['title'])  
    print('Author:', book[0].text)  
    print('Year: ', book[1].text, '\n')
```



My books:

Title: The Little Prince
Author: Antoine de Saint-Exupéry
Year: 1943

Title: Hamlet
Author: William Shakespeare
Year: 1603

FILE PROCESSING

- **xml.etree.ElementTree.**

- Búsqueda de elementos:

- Realizar búsqueda **recursiva** de elementos a partir del nombre la etiqueta con el método **iter**.

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')
```

```
root = tree.getroot()
```

```
for author in root.iter('author'):  
    print(author.text)
```



Antoine de Saint-Exupéry
William Shakespeare

FILE PROCESSING

- **xml.etree.ElementTree.**

- **Búsqueda de elementos:**

- Realizar búsqueda **en el primer nivel de profundidad** de elementos con el método **findall**.
- El método **findall** acepta expresiones XPath

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()  
for author in root.findall('author'):  
    print(author.text)
```



No proporciona resultados

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()  
for book in root.findall('book'):  
    print(book.get('title'))
```



The Little Prince
Hamlet

El método get permite acceder a los valores de los atributos

FILE PROCESSING

- **xml.etree.ElementTree.**

- **Búsqueda de elementos:**

- Realizar búsqueda **en el primer nivel de profundidad** del primer elemento con el método **find**.
- El método **find** acepta expresiones XPath

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()  
print(root.find('book').get('title'))
```



The Little Prince

FILE PROCESSING

- **xml.etree.ElementTree.**
 - **Modificación** de elementos:
 - Asignando un nuevo valor al atributo **tag** del elemento.

```
import xml.etree.ElementTree as ET

tree = ET.parse('books.xml')
root = tree.getroot()
for child in root:
    child.tag = 'movie'
    print(child.tag, child.attrib)
    for sub_child in child:
        print(sub_child.tag, ': ', sub_child.text)
```



```
movie {'title': 'The Little Prince'}
author : Antoine de Saint-Exupéry
year : 1943
movie {'title': 'Hamlet'}
author : William Shakespeare
year : 1603
```

FILE PROCESSING

- **xml.etree.ElementTree.**
 - **Eliminación** de elementos:
 - Utilizando el método **remove**.

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('books.xml')  
root = tree.getroot()
```

```
for child in root:  
    child.tag = 'movie'  
    child.remove(child.find('author'))  
    child.remove(child.find('year'))  
    print(child.tag, child.attrib)  
    for sub_child in child:  
        print(sub_child.tag, ': ', sub_child.text)
```



```
movie {'title': 'The Little Prince'}  
movie {'title': 'Hamlet'}
```

FILE PROCESSING

- **xml.etree.ElementTree.**
 - **Creación/modificación** de atributos:
 - Utilizando el método **set**.

```
import xml.etree.ElementTree as ET

tree = ET.parse('books.xml')
root = tree.getroot()
for child in root:
    child.tag = 'movie'
    child.remove(child.find('author'))
    child.remove(child.find('year'))
    child.set('rate', '5')
    print(child.tag, child.attrib)
    for sub_child in child:
        print(sub_child.tag, ': ', sub_child.text)
```



```
movie {'title': 'The Little Prince', 'rate': '5'}
movie {'title': 'Hamlet', 'rate': '5'}
```

FILE PROCESSING

- **xml.etree.ElementTree.**

- **Almacenamiento** de documento:

- Utilizando el método **write** de ElementTree.
 - Parámetros: nombre de fichero, sistema de codificación, próloto (True/False)

```
import xml.etree.ElementTree as ET

tree = ET.parse('books.xml')
root = tree.getroot()
for child in root:
    child.tag = 'movie'
    child.remove(child.find('author'))
    child.remove(child.find('year'))
    child.set('rate', '5')
    print(child.tag, child.attrib)
    for sub_child in child:
        print(sub_child.tag, ': ', sub_child.text)

tree.write('movies.xml', 'UTF-8', True)
```

FILE PROCESSING

- **xml.etree.ElementTree.**

- **Creación** de documento:

- Método **Element** de ElementTree para crear el elemento raíz.
 - Método **SubElement** de ElementTree para crear subelementos.
 - Método/constructor **ElementTree** para construir un ElementTree a partir del elemento raíz.

- La función **dump** de ElementTree se utiliza para depuración y muestra el documento XML por consola.

```
import xml.etree.ElementTree as ET

root = ET.Element('videojuegos')
movie_1 = ET.SubElement(root, 'videojuego', {'titulo': 'God of War', 'plataforma': 'PlayStation'})
movie_2 = ET.SubElement(root, 'videojuego', {'titulo': 'Gears of War', 'plataforma': 'Xbox'})
ET.dump(root)

tree = ET.ElementTree(root)
tree.write('games.xml', 'UTF-8', True)
```