Web Scraping en Python: HTML y BeautifulSoup (I)

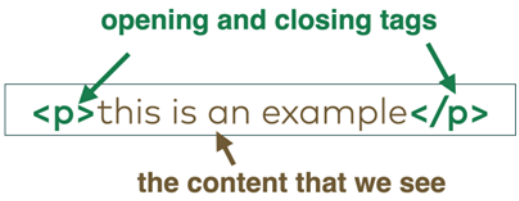
En esta sesión vamos a revisar unos cuantos conceptos importantes de HTML (el lenguaje en el que están construidas las páginas Web) pero lo haremos a la vez que vamos aprendiendo algo más de cómo usar BeautifulSoup. Para ello lo primero es hacer unas cuantas importaciones y crearnos una página HTML ficticia que nos servirá de guía:

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

# Suponemos que esta pagina de ejemplo la hemos descargado con una llamada a la función request de la librería requests
contenido = """
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Página de prueba</title>
</head>
<body>
<div id="main" class="full-width">
   <h1>El título de la página</h1>
   <p>Este es el primer párrafo</p>
   <p>Este es el segundo párrafo</p>
   <div id="innerDiv">
      <div class="links">
         <a href="https://pagina1.xyz/">Enlace 1</a>
         <a href="https://pagina2.xyz/">Enlace 2</a>
      </div>
      <div class="right">
         <div class="links">
            <a href="https://pagina3.xyz/">Enlace 3</a>
            <a href="https://pagina4.xyz/">Enlace 4</a>
         </div>
      </div>
   </div>
   <div id="footer">
      <!-- El footer -->
      <p>Este párrafo está en el footer</p>
      <div class="links footer-links">
         <a href="https://pagina5.xyz/">Enlace 5</a>
      </div>
   </div>
</div>
</body>
</html>
"""
soup = BeautifulSoup(contenido, 'lxml')
```

Primeros conceptos: Etiquetas y Atributos

HTML consta de elementos como enlaces, párrafos, encabezados, bloques, etc. Estos elementos están envueltos entre etiquetas; dentro de la etiqueta de apertura y cierre se puede encontrar el contenido del elemento.



Esto equivale a las etiquetas y nodos de XML. La diferencia es que muchas de las etiquetas de HTML tienen un significado predefinido (por ejemplo, p corresponde a paragraph).

Los elementos HTML también pueden tener atributos que contienen información adicional sobre el elemento. Los atributos se definen en las etiquetas de apertura con la siguiente sintaxis: nombre del atributo = "valor del atributo".

Son análogos a como se declaran en XML (no en vano, este es una extensión de HTML).

Nuestro cometido inicial en el Web Scraping será detectar que tags o etiquetas contienen nuestros datos y en que parte de esa etiqueta (en el atributo o en sus etiquetas hijas) está la información.

Por ejemplo, si te fijas en la "página" web ejemplo, tenemos etiquetas como: head, title, body, div, h1, p y a.

- Claramente p son párrafos de texto, h1 sabemos que es un header y a es el tag para links.
- Los tags a tienen un atributo que es href y este tiene como valor el link al que dirigen.
- A igual que XML, los tags tienen también un texto (el contenido que va entre la declaración del tag y su cierre por ejemplo en Enlace 1, es Enlace 1, va entre <a ...> y)

Y así con el resto.

¿Qué vamos a hacer nosotros? En un par de píldoras "inspeccionaremos" una página web hasta encontrar los tags que nos interesen y la parte de estos (atributos, texto) y luego cargaremos la página en una variable python y la procesaremos con BeautifulSoup, y lo usaremos para extraer la info.

Veamos ahora algo más de BeautifulSoup a partir de la página ficticia

BeautifulSoap: Tags y NavigableString

Tenemos principalmente 2 tipos de objeto BeautifulSoup:

Tag

- Tag: Este objeto se corresponde con una etiqueta HTML o XML. Por ejemplo, dado el objeto soup, podemos acceder al objeto (tag) que representa al título de la página usando la etiqueta title.

soup.title

<title>Página de prueba</title>

Hemos accedido a todo el objeto (su apertura, cierre, texto, atributos, etc). Probemos con otro objeto, por ejemplo: div

soup.div

<h1>El título de la página</h1>
Este es el primer párrafo
Este es el segundo párrafo

Enlace 1
Enlace 2

Enlace 3
Enlace 4

<!-- El footer -->
Este párrafo está en el footer

Enlace 5

Como hay varios objetos tipo nos ha devuelto toda la definición del primero que encuentra (que es el div de mayor extensión que contiene al resto) (la etiquta div es muy utilizada para hacer agrupaciones de objetos a los que luego aplicar estilos comunes a partir de hojas css, y demás...) aquí tienes una guía de las miles que hay en Internet.

 # Veamos como acceder a uno de los div internos
soup.div.div

Enlace 1

```
<a href="https://pagina2.xyz/">Enlace 2</a>
</div>
<div class="right">
<div class="links">
<a href="https://pagina3.xyz/">Enlace 3</a>
<a href="https://pagina4.xyz/">Enlace 4</a>
</div>
</div>
</div>
```

Así es un poco difícil de seguir un árbol HTML, y eso que en nuestro caso es una página sencilla. **BeautifulSoup tiene un método para pintar de forma tabulada un árbol: prettify** (que puedes aplicar a cualquier objeto Tag)

```
print(soup.div.div.prettify())
```

```
<div id="innerDiv">
 <div class="links">
  <a href="https://pagina1.xyz/">
   Enlace 1
  </a>
  <a href="https://pagina2.xyz/">
   Enlace 2
  </a>
 </div>
 <div class="right">
  <div class="links">
   <a href="https://pagina3.xyz/">
    Enlace 3
   </a>
   <a href="https://pagina4.xyz/">
    Enlace 4
   </a>
  </div>
 </div>
</div>
```

Y para todo el árbol:

```
print(soup.prettify())
```

```
<html lang="es">
 <head>
  <meta charset="utf-8"/>
  <title>
   Página de prueba
  </title>
 </head>
 <body>
```

```
 <div class="full-width" id="main">
  <h1>
   El título de la página
  </h1>
  <p>
   Este es el primer párrafo
  </p>
  <p>
   Este es el segundo párrafo
  </p>
  <div id="innerDiv">
   <div class="links">
    <a href="https://pagina1.xyz/">
     Enlace 1
    </a>
    <a href="https://pagina2.xyz/">
     Enlace 2
    </a>
   </div>
   <div class="right">
    <div class="links">
     <a href="https://pagina3.xyz/">
      Enlace 3
     </a>
     <a href="https://pagina4.xyz/">
      Enlace 4
     </a>
    </div>
   </div>
  </div>
  <div id="footer">
   <!-- El footer -->
   <p>
    Este párrafo está en el footer
   </p>
   <div class="links footer-links">
    <a href="https://pagina5.xyz/">
     Enlace 5
    </a>
   </div>
  </div>
 </div>
</body>
</html>
```

Dado un objeto de tipo Tag, **podemos acceder a sus atributos tratando al objeto como si fuera un diccionario**. Además, se puede acceder a ese diccionario por medio del atributo attrs:

 # Por ejemplo el primer gran div tiene dos atributos (si te fijas en la salida de la celda de código anterior)

```
soup.div
```

```
<div class="full-width" id="main">
<h1>El título de la página</h1>
<p>Este es el primer párrafo</p>
<p>Este es el segundo párrafo</p>
<div id="innerDiv">
<div class="links">
<a href="https://pagina1.xyz/">Enlace 1</a>
<a href="https://pagina2.xyz/">Enlace 2</a>
</div>
<div class="right">
<div class="links">
<a href="https://pagina3.xyz/">Enlace 3</a>
<a href="https://pagina4.xyz/">Enlace 4</a>
</div>
</div>
</div>
<div id="footer">
<!-- El footer -->
<p>Este párrafo está en el footer</p>
<div class="links footer-links">
<a href="https://pagina5.xyz/">Enlace 5</a>
</div>
</div>
</div>
```

```
print("Id:", soup.div["id"])
Id: main
```

```
print("Class:", soup.div["class"])
print(soup.div.attrs)
Class: ['full-width']
{'id': 'main', 'class': ['full-width']}
```

Igual has caído en la cuenta de que cuando usábamos ElementTree con XML el atributo (valga la redundancia) que contenía los atributos de un nodo era attrib y con BSoup es attrs.

NavigableString y text/get_text()

- **NavigableString**: Un objeto de este tipo representa a la cadena de texto que hay contenida en una etiqueta. Se accede por medio de la propiedad string.

```
primer_parrafo = soup.p
print(primer_parrafo.string)
print(type(primer_parrafo.string))
```

```
Este es el primer párrafo

<class 'bs4.element.NavigableString'>
```

- **text**: Cuando un nodo tiene más de un subnodo con texto string devuelve "None", es por eso que, si sabes que ese es el caso, es mejor utilizar el atributo text, ojo que devuelve todos los textos internos:

```
print(soup.div.string)
None

print(soup.div.text)
El título de la página
Este es el primer párrafo
Este es el segundo párrafo

Enlace 1
Enlace 2

Enlace 3
Enlace 4

Este párrafo está en el footer

Enlace 5


print(soup.div.get_text(separator = "*"))
*El título de la página*
*Este es el primer párrafo*
*Este es el segundo párrafo*
*
*Enlace 1*
*Enlace 2*
*
*Enlace 3*
*Enlace 4*
*
*Este párrafo está en el footer*
*
*Enlace 5*
*
```

Nosotros vamos a querer capturar atributos o textos (navigable strings), pero claro para ello necesitamos además poder navegar por la estructura y eso es lo que veremos cómo hacer en la siguiente sesión.