Ejercicios:

1.- Revisión de productos

Una tarea muy común de los web crawlers es la de verificar y consultar datos en las galerías de productos de tiendas online. Por ejemplo, podríamos crear un crawler que revise aquellos productos que no estén en stock (en el código HTML aparecerá algún aviso para que se vea en el navegador) para así poder jugar con los precios de los mismos productos en nuestra página web. Esta tarea puede ser muy tediosa pero vamos a implementar con BeautifulSoup un programa que lleve más o menos la misma filosofía pero a mucha menor escala:

Entre en la web www.demartina.com y elija alguna galería de productos (http://www.demartina.com/lego-star-wars-c-73_29_50.html por ejemplo). Estudie el código HTML de la galería desde el navegador e imprima el nombre, la referencia y el/los precio/precios de todos los productos. Algunos precios pueden tener un descuento, considere el precio nuevo y el antiguo y muestre el descuento aplicado. Evite aquellos productos que estén envueltos en un 'div' con clase CSS 'blueknow-item' ya que se generan automáticamente y no se añaden al DOM de la misma forma.

2.- Información jornada Liga BBVA.

Un uso que podemos darle a BeautifulSoup es el de crearnos una consola con la que personalizar las búsquedas de información y no tener que visitar varias webs y tener muchas pestañas abiertas en el navegador. La idea es la de tener cargado en nuestro programa las URL de las páginas webs que nos interesen y mediante los parámetros de entrada hacer la búsqueda pertinente. Vamos a hacer un pequeño ejemplo:

En las páginas web de los diarios deportivos, suelen poner una retransmisión en directo de cada partido de fútbol con el minuto y la acción ocurrida (una tabla, en definitiva). Se pide realizar un crawler haciendo uso de BeautifulSoup que a raíz del número de la jornada (de la 1 a la 38) imprima los resultados de cada partido así como el minuto y la acción de cada gol. Para ello siga por ejemplo la web http://www.as.com/resultados/futbol/primera/. Dentro de cada resultado, hay un link hacia la web de la retransmisión en directo de la que puede tomar los datos a imprimir.

3.- Información de stock.

Volviendo al mundo de las compras online y de los stock de los productos, otro uso que podríamos darle a BeautifulSoup es la de avisar cuando un producto esté disponible (en caso que la empresa no se encargue de hacerlo). En el fondo, la técnica que se utiliza para llevar a cabo esta tarea se puede llevar a cualquier ámbito en el que lo que necesitemos sea detectar un cambio en la información que se muestra al usuario (una resolución, una publicación, una convocatoria, etc.). La idea es tener el programa pendiente de algún cambio y que cuando se produzca deje de revisar el código. Esta solución no es muy óptima ya que implica tener el proceso en una espera activa (polling) revisando código una y otra vez, pero en el ámbito de aprendizaje en el que nos movemos es suficiente...

Por ejemplo, puede acceder a la web de Google donde venden el smartphone Nexus 4 (https://play.google.com/store/devices/details?id=nexus_4_8gb&feature=microsite&hl=es),revisar

donde se almacena la información del stock y hacer un bucle donde se verifique que esa información no ha cambiado y que cuando cambie, salga. El tratamiento una vez que se obtiene el cambio que se desea es libre: mandar un e-mail (SMTP), imprimir un mensaje, etc.

4.- Noticias.

Aparte de código HTML, BeautifulSoup es capaz de hacer análisis sintáctico de otros lenguajes basados en etiquetas como XML. El funcionamiento es similar (tendríamos que indicarle que use 'lxml' como analizador sintáctico a la hora de crear el árbol), tendremos etiquetas con atributos y la misma relación entre nodos hermanos y nodos padres que tenemos en los árboles que se generan con código HTML.

XML es el formato usado para la difusión de noticias de forma actualizada (RSS) en la red usando 'agregadores'. RSS 2.0 sigue una especificación en la que se define la forma en la que una noticia es representada:

```
<rss version="2.0">
<channel>
        <title>
        <description>
        <link>
        <item> … </item>
        <item> … </item>
</channel>
</rss>
```

Siendo cada <item> una noticia de las que se ofrecen en ese archivo .xml.

Para este ejercicio, realizaremos un pequeño buscador de noticias. A partir de varias fuentes de RSS (de cualquier diario, por ejemplo: http://rss.elmundo.es/rss/, http://servicios.elpais.com/rss/, http://www.abc.es/rss), a poder ser de la misma categoría, realize un programa el cual pida una cadena de búsqueda y busque las coincidencias en el texto de cada <item> (en el <title> y <description> de la noticia por ejemplo) y las devuelva clasificadas por la fuente donde aparecen. En la web http://www.rssboard.org/rss-specification tiene la especificación de los documentos XML para RSS en caso de que necesite ayuda para entender los archivos. Se valorará la inclusión de una interfaz gráfica en la que sea más cómodo hacer la búsqueda.

5.- Modificación de atributos.

Estando realizando sus prácticas en empresa, su jefe le comunica que las imágenes que se muestran en la página web van a ser migradas a otra localización dentro del servidor, lo que implica que hay que modificar todos los atributos 'src' de las etiquetas <img> para que muestren las mismas imágenes en la localización nueva. Tras varios intentos (cabezazos) usando PHP y expresiones regulares, se da cuenta de que sería más inteligente usar BeautifulSoup y Python para analizar sintácticamente los documentos HTML rebajando el uso de expresiones regulares. El cambio de localización en el servidor se interpreta como cambiar la ruta que aparece en 'src', por ejemplo:

Cambiaría a:

En este caso, aunque mínima, haría falta usar una expresión regular.

Se pide en el ejercicio coger alguna página web donde se muestre alguna galería de fotos medianamente extensa (The Big Picture de www.boston.com, por ejemplo http://www.boston.com/bigpicture/2013/05/national_geographic_traveler_m_1.html), estudiar las etiquetas <img> y decidir hacer algún cambio en el atributo (en las galerías de The Big Picture por ejemplo, eliminar /site_graphics/ de la URL y añadir /new_site/ en su lugar por ejemplo).

6.- Creación de un documento HTML desde cero.

Una utilidad de la que dispone BeautifulSoup es la de crear etiquetas nuevas desde cero (soup.new_tag()) y poder usarlas para añadirlas en el árbol original, con el propósito de almacenarlo en un nuevo fichero HTML con los cambios aplicados.

Imagínese que ha terminado trabajando como ingeniero informático en una central nuclear. Esta central nuclear posee un reactor modelo HTR-200, el cual puede alcanzar temperaturas de más de 1000ºC. Su misión consiste en mantener el sistema web existente en la red interna de la central en la que se monitoriza el estado del reactor. Una de las utilidades que debe implementar es imprimir en una tabla una serie de valores de temperatura tomados directamente del reactor: en color rojo si el valor es peligroso,verde si es estable o azul si necesita más temperatura. El tamaño de datos que debe tratar es muy grande (hablamos de miles de datos) y estar escribiendo uno a uno en un <td> dentro de un <tr> dentro de un <table> es cuanto menos, tedioso (y aburrido). Para solucionar este problema, tiene la siguiente idea:

Realizar un programa en Python usando BeautifulSoup el cual reciba todos los valores en una lista y, conforme vaya iterando sobre ella,vaya formateando la tabla en HTML y modificando el color en el que será almacenado dentro de la tabla . Recuerde que debe partir desde un código HTML vacío e ir añadiendo las etiquetas al <body> conforme se vayan completando. Aquellos valores que superen 1200ºC, irán en rojo; aquellos que se mantengan por debajo hasta 750ºC, irán de verde, por debajo de 750ºC deberán ir en azul. Una vez que tenga la tabla completada, guárdela en un archivo y mire los resultados. Se le incluye el código que simula la toma de datos: pide por consola el número de datos totales y el número de filas y de columnas en los que desea imprimirlos. Pista: estos dos últimos valores serán sobre los que deberá iterar.