Scraping

Requêtes avec le package `requests`

Entrée [1]:

```
import requests
```

Entrée [2]:

```
page = requests.get("http://dataquestio.github.io/web-scraping-pages/simple.html")
```

Entrée [4]:

```
help(page)
```

```
 |      been processed automatically (by `Session.resolve_redirec
ts`).
 |
 |  links
 |      Returns the parsed header links of the response, if any.
 |
 |  next
 |      Returns a PreparedRequest for the next request in a redirect ch
ain, if there is one.
 |
 |  ok
 |      Returns True if :attr:`status_code` is less than 400, False if
not.
 |
 |      This attribute checks if the status code of the response is bet
ween
 |      400 and 600 to see if there was a client error or a server erro
r. If
 |      the status code is between 200 and 400, this will return True.
This
```

Entrée [5]:

```
page.status_code
```

Out[5]:

```
200
```

Entrée [6]:

```
page.content
```

Out[6]:

```
b'<!DOCTYPE html>\n<html>\n    <head>\n        <title>A simple example pag
e</title>\n    </head>\n    <body>\n        <p>Here is some simple content
for this page.</p>\n    </body>\n</html>'
```

Pour une page si simple, on pourrait presque faire le parsing et extraire l'information manuellement :

Entrée [7]:

```
page.text.split("\n")
```

Out[7]:

```
['<!DOCTYPE html>',
 '<html>',
 '    <head>',
 '        <title>A simple example page</title>',
 '    </head>',
 '    <body>',
 '        <p>Here is some simple content for this page.</p>',
 '    </body>',
 '</html>']
```

On obtient le contenu dans différents éléments (tags), que l'on peut sélectionner - mais difficile de rechercher précisément ce que l'on veut là-dedans ! `beautifulsoup` va nous aider.

Parsing avec `BeautifulSoup`

Entrée [8]:

```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\users\azade\anaconda3\lib\site-pa
ckages (0.0.1)
Requirement already satisfied: beautifulsoup4 in c:\users\azade\anaconda3
\lib\site-packages (from bs4) (4.10.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\azade\anaconda3\l
ib\site-packages (from beautifulsoup4->bs4) (2.2.1)
```

Entrée [9]:

```
from bs4 import BeautifulSoup
```

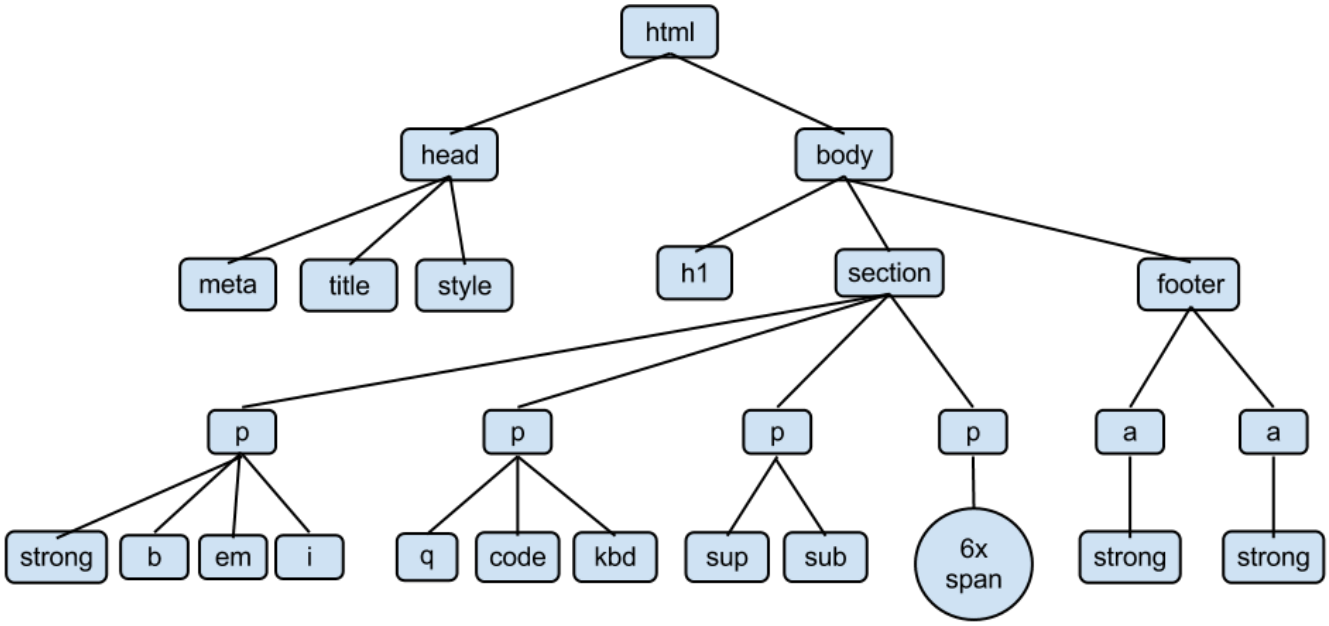
Entrée [10]:

```
# on fournit à BeautifulSoup le contenu brut de la page (HTML)
soup = BeautifulSoup(page.content)
soup
```

Out[10]:

```
<!DOCTYPE html>
<html>
<head>
<title>A simple example page</title>
</head>
<body>
<p>Here is some simple content for this page.</p>
</body>
</html>
```

Visiblement, rien de nouveau. Pourtant, les tags sont désormais organisés en hiérachie, ou en **arbre**, comme vu dans le cours théorique ! Chaque tag est appelé **node (noeud)**, et est situé à un certain niveau dans l'arbre. Le noeud le plus haut est `<html>`, puis suivent ses **children** `<head>` et `<body>`, au même niveau.



A l'intérieur de **`<head>`** se trouvent des informations sur la page qui ne sont pas visibles quand la page est affichée dans un navigateur.

A l'intérieur de **`<body>`** se trouve le contenu visible pour les utilisateurs.

Les enfants de `<body>` sont **`<h1>`** (un titre), `<section>` (littéralement : une section qui contient des paragraphes de texte **`<p>`**), et un `<footer>` (qui contient des liens **`<a>`**).

Comment rechercher une information dans cette hiérarchie (cette soupe !) ?

Entrée [11]:

```
# Bonus : bs dispose d'une méthode pour afficher le HTML de façon bien indentée
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   A simple example page
  </title>
 </head>
 <body>
  <p>
   Here is some simple content for this page.
  </p>
 </body>
</html>
```

Entrée [ ]: