Scraping

Entrée [1]:

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

Entrée [2]:

```
# Requête et parsing de la page souhaitée
# Nous chargeons ici une page un peu plus compliquée
page = requests.get("http://dataquestio.github.io/web-scraping-pages/ids_and_classes.htm
soup = BeautifulSoup(page.content)
soup
```

Out[2]:

```
<html>
<head>
<title>A simple example page</title>
</head>
<body>
<div>
<p class="inner-text first-item" id="first">
                First paragraph.
            </p>
<p class="inner-text">
                Second paragraph.
            </p>
</div>
<p class="outer-text first-item" id="second">
<b>
                First outer paragraph.
            </b>
</p>
<p class="outer-text">
<b>
                Second outer paragraph.
            </b>
</p>
</body>
</html>
```

Entrée [5]:

```
print(page)
```

```
<Response [200]>
```

Entrée [3]:

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

```
<html>
 <head>
  <title>
   A simple example page
  </title>
 </head>
 <body>
  <div>
   <p class="inner-text first-item" id="first">
    First paragraph.
   </p>
   <p class="inner-text">
    Second paragraph.
   </p>
  </div>
  <p class="outer-text first-item" id="second">
   <b>
    First outer paragraph.
   </b>
  </p>
  <p class="outer-text">
   <b>
    Second outer paragraph.
   </b>
  </p>
 </body>
</html>
```

Trouver des éléments grâce à leur tag

La grande force de Beautiful Soup est la recherche de contenu. Nous pouvons extraire un élément grâce à son **tag** avec la méthode **`.find('tag')`** :

Entrée [6]:

```
soup.find('p')
```

Out[6]:

```
<p class="inner-text first-item" id="first">
               First paragraph.
           </p>
```

Attention : `find` ne récupère que le premier tag qui remplit la condition. Pour obtenir tous les tags de la page remplissant la condition, il faut utiliser **`find_all`** . Le résultat est sous forme de **liste** :

Entrée [5]:

```
soup.find_all('p')
```

Out[5]:

```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>,
 <p class="inner-text">
                Second paragraph.
            </p>,
 <p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>,
 <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

Une fois que le tag souhaité est isolé, `bs` nous permet d'extraire son contenu avec `.get_text()` :

Entrée [6]:

```
soup.find('p').get_text()
```

Out[6]:

```
'\n                First paragraph.\n            '
```

Entrée [9]:

```
# Bonus : .strip() pour enlever les blancs avant et après le texte extrait
soup.find('p').get_text().strip()
```

Out[9]:

```
'First paragraph.'
```

Entrée [7]:

```
# Attention : .get_text ne fonctionne pas sur une liste ('find_all')
soup.find_all('p').get_text()
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp/ipykernel_13328/3412720215.py in <module>
      1 # Attention : .get_text ne fonctionne pas sur une liste ('find_al
l')
----> 2 soup.find_all('p').get_text()

~\anaconda3\lib\site-packages\bs4\element.py in __getattr__(self, key)
   2251     def __getattr__(self, key):
   2252         """Raise a helpful exception to explain a common code fi
x."""
-> 2253         raise AttributeError(
   2254             "ResultSet object has no attribute '%s'. You're probab
ly treating a list of elements like a single element. Did you call find_al
l() when you meant to call find()?" % key
   2255         )

AttributeError: ResultSet object has no attribute 'get_text'. You're proba
bly treating a list of elements like a single element. Did you call find_a
ll() when you meant to call find()?
```

Entrée [11]:

```
soup.find_all('p')
```

Out[11]:

```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>,
 <p class="inner-text">
                Second paragraph.
            </p>,
 <p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>,
 <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

Entrée [6]:

```
# Nous pouvons utiliser une liste en compréhension
texte = [ p.get_text().strip() for p in soup.find_all('p') ]
texte
```

Out[6]:

```
['First paragraph.',
 'Second paragraph.',
 'First outer paragraph.',
 'Second outer paragraph.']
```

Plus précis : trouver des éléments grâce à leurs propriétés

Comment faire pour extraire seulement le second paragraphe ?

Nous allons utiliser ici les **"attributs"** de chaque tag.

Ici, tous nos paragraphes `<p>` ont une **`class`** et un **`id`** . Ces attributs sont utilisés pour la mise en page du site, mais nous allons les utiliser pour notre recherche !

Entrée [15]:

```
soup.find_all('p', {"class":"outer-text"})
```

Out[15]:

```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>,
 <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

Entrée [17]:

```
soup.find_all('p', {"class":"inner-text"})
```

Out[17]:

```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>,
 <p class="inner-text">
                Second paragraph.
            </p>]
```

Entrée [14]:

```
soup.find_all('p', {"class":"outer-text", "id":"second"})
```

Out[14]:

```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>]
```

Entrée [9]:

```
soup.find_all('p', {"class":"inner-text", "id":"first"})
```

Out[9]:

```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>]
```

Entrée [10]:

```
# BONUS : un id étant unique en HTML, on peut y accéder directement sans mentionner le t
soup.find(id="first")
```

Out[10]:

```
<p class="inner-text first-item" id="first">
                First paragraph.
            </p>
```

→ Avec la méthode **`find_all`** et un **dictionnaire d'attributs** (`class` et `id` sont les plus communs, mais il en existe beaucoup d'autres selon le site scrapé), vous allez pouvoir précisément **trouver** le tag souhaité et **extraire** son contenu.

Entrée [ ]:

```

```