Web scraping avec Python

---

source : revue Linux Références, hors-série n°8, "Spécial Python", article : "Extraire automatiquement des données du web avec Pyhton", p.44-49.

Télécharger une page web

In [30]:
```
import requests

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

Out[30]:
```
<Response [200]>
```

Un code 200 indique que la page a été téléchargée avec succès.

Afficher le contenu d'une page web

In [32]:
```
page.content
```

Out[32]:
```
b'<!DOCTYPE html>\n<html>\n    <head>\n        <title>A simple example page</title
>\n    </head>\n    <body>\n        <p>Here is some simple content for this page.
</p>\n    </body>\n</html>'
```

Analyse avec BeautifuLSoup

In [33]:
```
from bs4 import BeautifulSoup

soup = BeautifulSoup (page.content, 'html.parser')
print(soup)
```

```
<!DOCTYPE html>

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

Affiché le contenu HTML formaté

In [34]:
```
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>
```

On peut sélectionner tous les éléments au niveau supérieur de la page avec la propriété `children soup` . Comme elle renvoie un générateur de liste, il faut appler la fonction `list` :

In [35]:
```
soup.children
```

Out[35]:
```
<list_iterator at 0x249311f8910>
```

In [36]:
```
list(soup.children)
```

Out[36]:
```
['html',
 '\n',
 <html>
 <head>
 <title>A simple example page</title>
 </head>
 <body>
 <p>Here is some simple content for this page.</p>
 </body>
 </html>]
```

Type de chaque élément dans la liste :

In [37]:
```
[type(element) for element in list(soup.children)]
```

Out[37]:
```
[bs4.element.Doctype, bs4.element.NavigableString, bs4.element.Tag]
```

Tous les éléments sont des objets **Beautiful Soup**.

Le premier est un objet **Doctype** : il contient des informations sur le type du document.

Le deuxième est un objet de type **NavigableString**, qui représente le texte trouvé dans le document HTML.

Le troisième est un objet **Tag**, qui contient d'autres balises imbriquées. C'est le type d'objet le plus important et le plus utilisé en Web scraping.

L'objet **Tag** permet de naviguer dans un document HTML et d'extraire d'autres balises et du texte. On peut donc sélectionner la balise html et ses enfants (children) en choisissant ce dernier élément de la liste :

In [71]:
```
html = list(soup.children)[2]
html
```

Out[71]:
```
<html>
<head>
<title>A simple example page</title>
</head>
<body>
<p>Here is some simple content for this page.</p>
</body>
</html>
```

Chaque élément de la liste retourné par la propriété **children** étant un objet **BeautifulSoup**, il est possible d'appeler la méthode **children** sur le bloc **html** :

In [72]:
```
list(html.children)
```

Out[72]:
```
['\n',
 <head>
 <title>A simple example page</title>
 </head>,
 '\n',
 <body>
 <p>Here is some simple content for this page.</p>
 </body>,
 '\n']
```

Il y a 2 balises : **head** et **body**. Pour extraire le texte à l'intérieur de la balise **p**, il faut utiliser le bloc **body** :

In [74]:
```
body = list(html.children)[3]
body
```

Out[74]:
```
<body>
<p>Here is some simple content for this page.</p>
</body>
```

On cherche les enfants de la balise **body** :

In [75]:
```
list(body.children)
```

Out[75]:
```
['\n', <p>Here is some simple content for this page.</p>, '\n']
```

On peut maintenant isoler la balise **p**:

In [81]:
```
p = list(body.children)[1]
p
```

Out[81]:
```
<p>Here is some simple content for this page.</p>
```

Une fois isolée, on utilise la méthode **get_text** pour extraire le texte à l'intérieur de la balise :

In [82]:
```
p.get_text()
```

Out[82]:
```
'Here is some simple content for this page.'
```

Autre méthode :

In [86]:
```
enfants= html.findChildren()
enfants
```

Out[86]:
```
[<head>
 <title>A simple example page</title>
 </head>,
 <title>A simple example page</title>,
 <body>
 <p>Here is some simple content for this page.</p>
 </body>,
 <p>Here is some simple content for this page.</p>]
```

In [78]:
```
# On sélectionne le body :
enfants[2]
```

Out[78]:
```
<body>
<p>Here is some simple content for this page.</p>
</body>
```

In [61]:
```
# On sélectionne la balise 'p':
enfants[3]
```

Out[61]:
```
<p>Here is some simple content for this page.</p>
```

In [83]:
```
# on utilise get_text():
enfants[3].get_text()
```

Out[83]:
```
'Here is some simple content for this page.'
```

Recherche de toutes les instances d'un tag :

La méthode la plus simple et la plus rapide pour extraire une seule balise est la méthode **find_all**, qui trouve toutes les instances d'une balise sur une page.

In [90]:
```
soup = BeautifulSoup (page.content, 'html.parser')
soup.find_all('p')
```

Out[90]:
```
[<p>Here is some simple content for this page.</p>]
```

Comme **find_all** renvoie une liste, on peut utiliser l'indexation de liste pour extraire le texte, toujours avec la méthode **get_text()** :

In [92]:
```
soup.find_all('p')[0].get_text()
```

Out[92]:
```
'Here is some simple content for this page.'
```

Si on veut rechercher uniquement la première instance d'une balise, il est possible d'utiliser la méthode **find**, qui renverra un seul objet **BeautifulSoup** :

In [95]:
```
soup.find('p')
```

Out[95]:
```
<p>Here is some simple content for this page.</p>
```

Recherche de tags par classe et identifiant

Les classes et les identifiants sont utilisés par CSS pour déterminer à quels éléments HTML appliquer certains styles. On peut ainsi préciser quels éléments spécifiques doivent être récupérés.

In [100…
```
page = requests.get("http://dataquestio.github.io/web-scraping-pages/ids_and_classe
soup = BeautifulSoup(page.content, 'html.parser')
soup
```

Out[100]:
```
<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>
```

Nous pouvons utiliser la méthode **find_all** pour rechercher des éléments par classe/identifiant. Ici, nous recherchons toute balise **p** contenant la classe **outer-text**:

In [101…
```
soup.find_all('p', class_='outer-text')
```

Out[101]:
```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>,
 <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

Nous pouvons rechercher maintenant **toute** balise contenant la classe **outer-text** :

In [103…
```
soup.find_all(class_='inner-text')
```

Out[103]:
```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>,
 <p class="inner-text">
                Second paragraph.
            </p>]
```

On peut également rechercher des éléments par identifiants :

In [104…
```
soup.find_all(id="first")
```

Out[104]:
```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>]
```

In [105…
```
soup.find_all(id='second')
```

Out[105]:
```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>]
```

Utilisation des sélecteurs CSS

On peut rechercher des éléments à l'aide des sélecteurs CSS. Ces sélecteurs permettent au langage CSS de spécifier des balises HTML à styliser.

- **p a** : trouve toutes les balises **a** à l'intérieur d'une balise **body**.
- **body p a** : trouve toutes les balises à l'intérieur d'une balise **p** à l'intérieur d'une balise **body**.
- **html body** : trouve toutes les balises **body** à l'intérieur d'une balise **html**.
- **p.nom_classe** : trouve toutes les balises **p** avec une classe spécifiée.
- **p # identifiant** : trouve toutes les balises **p** avec un identifiant spécifié.
- **body.p.nom_class** : trouve toutes les balises **p** avec une classe spécifiée à l'intérieur d'une balise **body**.

Téléchargement des données météo

In [121…
```
page = requests.get("http://forecast.weather.gov/MapClick.php?lat=37.7772&lon=-122
soup = BeautifulSoup(page.content, 'html.parser')

'''Trouvons la div avec l'id de prévision de 7 jours et attribuons la à la variable
A l'intérieur de seven_day nous recherchons chaque élément de prévision individuel

seven_day = soup.find(id="seven-day-forecast")

previsions_items = seven_day.find_all(class_="tombstone-container")

tonight = previsions_items[0]

print(tonight.prettify())
```

```
<div class="tombstone-container">
 <p class="period-name">
  Today
  <br/>
  <br/>
 </p>
 <p>
  <img alt="Today: Mostly cloudy, then gradually becoming sunny, with a high near
68. West wind 9 to 14 mph increasing to 16 to 21 mph in the afternoon. Winds could
gust as high as 30 mph. " class="forecast-icon" src="newimages/medium/bkn.png" tit
le="Today: Mostly cloudy, then gradually becoming sunny, with a high near 68. West
wind 9 to 14 mph increasing to 16 to 21 mph in the afternoon. Winds could gust as
high as 30 mph. "/>
 </p>
 <p class="short-desc">
  Decreasing
  <br/>
  Clouds
 </p>
 <p class="temp temp-high">
  High: 68 °F
 </p>
</div>
```

Extraire des informations

In [125…
```
# période : aujourd'hui
period = tonight.find(class_="period-name").get_text()

# courte description
short_desc = tonight.find(class_="short-desc").get_text()

# température (en Fahrenheit)
temp = tonight.find(class_="temp").get_text()

print(period)
print(short_desc)
print(temp)
```

```
Today
DecreasingClouds
High: 68 °F
```

In [126…
```
# extraction de 'title' dans la balise 'img' :
# On traite l'objet BeautifulSoup comme un dictionnaire, en passant l'attribut souh

img = tonight.find("img")
desc = img['title']
print(desc)
```

```
Today: Mostly cloudy, then gradually becoming sunny, with a high near 68. West win
d 9 to 14 mph increasing to 16 to 21 mph in the afternoon. Winds could gust as hig
h as 30 mph.
```

Construire un dataframe

Nous appellons la classe **DataFrame** et transmettre chaque liste d'éléments que nous avons.

Nous les transmettons dans le framework d'un dictionnaire.

Chaque clé de dictionnaire deviendra une colonne dans le **DataFrame**, et chaque liste deviendra les valeurs de la colonne.

In [142…

```
import pandas as pd

meteo = pd.DataFrame({
                      "periode": [period],
                      "short_desc": [short_desc],
                      "temp": [temp],
                      "desc": [desc]
                     })

meteo
```

Out[142]:

	periode	short_desc	temp	desc
0	Today	DecreasingClouds	High: 68 °F	Today: Mostly cloudy, then gradually becoming ...