DataScientest • com

Webscraping with Python

Collect data

Most websites are complex and have perpetual changes, making work of the scraper difficult.
The recovered HTML document is often obscure and the identification of the elements sought is not obvious. The `BeautifulSoup` module provides different methods to facilitate the search for data.
To avoid ending up with an obsolete code and non - usable work, it is important, once the elements have been identified and recovered, to save them in an external file.

The objectives of this notebook are as follows:

- Identify the elements in the tree using the attributes
- Recover the data in a dataframe and in a CSV file

Search for the elements

1. find, findAll, select

`BeautifulSoup` has several methods to select items from the HTML code.

- `find` :

  The `find` method is similar to navigation by tag. It allows you to recover the first tag concerned. Thus, the following two codes are equivalent:

  ```
  soup.div.div.a
  soup.find('div').find('div').find('a')
  ```

  They make it possible to display the first hypertext link of the HTML document `soup' hosted in two containers.

- `findAll` :

  This method recovers all the tags sought. For example, to display all hypertext links on the page, it is possible to use the following code:

  ```
  soup.findAll('a')
  ```

  The `findAll` function returns a **list** containing a series of values. The first value on the list is the `html` code of the first tag `a` of `soup` , the second value is the code of the second tag `a` of `soup` and so on.

- `select` :

  In addition to finding all the pointed elements, `select` has the particularity of being able to specify the parent of the tags concerned. For example, the code:

  ```
  soup.select("div > a")
  ```

  allows you to display all the tags `a` of the parent `div` (that is to say, locating just below the tag named `div`). This order also returns a **list** containing all the tags concerned.

- **a)** Import the modules necessary for web scraping.
- **b)** Recover the HTML code of the best Amazon books (https://www.amazon.fr/gp/bestsellers/books/301132/ref=zg_bs_nav_books_1) in a variable named `page` and create a `BeautifulSoup` object.
- **c)** Thanks to the techniques set out in the first notebook, identify the name of the tag containing the title of each book. Then, using the `findAll` method, display all the beacons of the same name.

In [5]:

```python
### Insert your code
from urllib.request import urlopen
from bs4 import BeautifulSoup as bs


url = "https://tr.wikipedia.org/wiki/Anasayfa"
page_wiki = urlopen(url)

soup = bs(page, 'html.parser')

soup.findAll('div')


amazon_url = "https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0"
page_amazon = urlopen(amazon_url)
soup = bs(page_amazon, 'html.parser')
soup.findAll('div')
```

Out[5]:

```
[<div id="a-page"><script data-a-state='{"key":"a-wlab-states"}' type="a-state">{"AUI_TNR_V2_180836":"C","AUI_ACCORDION_A11Y_RO
LE_354025":"T1","AUI_PRELOAD_261698":"C","AUI_TEMPLATE_WEBLAB_CACHE_333406":"C","AUI_72554":"C","AUI_KILLSWITCH_CSA_LOGGER_3729
63":"C","AUI_REL_NOREFERRER_NOOPENER_309527":"C","AUI_PCI_RISK_BANNER_210084":"C"}</script><script>typeof uex === 'function' &&
uex('ld', 'portal-bb', {wb: 1})</script><!-- sp:end-feature:start-body -->
 <!-- sp:feature:csm:body-open -->
 <!-- sp:end-feature:csm:body-open -->
 <!-- sp:feature:nav-inline-js -->
 <!-- NAVYAAN JS -->
 <script type="text/javascript">!function(n){function e(n,e){return{m:n,a:function(n){return[].slice.call(n)}(e)}}document.crea
teElement("header");var r=function(n){function u(n,r,u){n[u]=function(){a._replay.push(r.concat(e(u,arguments)))}}var a={};retu
rn a._sourceName=n,a._replay=[],a.getNow=function(n,e){return e},a.when=function(){var n=[e("when",arguments)],r={};return u(r,
n,"run"),u(r,n,"declare"),u(r,n,"publish"),u(r,n,"build"),r.depends=n,r.iff=function(){var r=n.concat([e("iff",arguments)]),a=
{};return u(a,r,"run"),u(a,r,"declare"),u(a,r,"publish"),u(a,r,"build"),a},r},u(a,[],"declare"),u(a,[],"build"),u(a,[],"publis
h"),u(a,[],"importEvent"),r._shims.push(a),a};r._shims=[],n.$Nav||(n.$Nav=r("rcx-nav")),n.$Nav.make||(n.$Nav.make=r)}(window)</
script><script type="text/javascript">
 $Nav.importEvent('navbarJS-beaconbelt');
 $Nav.declare('img.sprite', {
   'png32': 'https://m.media-amazon.com/images/G/35/gno/sprites/nav-sprite-global-1x-hm-dsk-reorg._CB405936783_.png',
   'png32-2x': 'https://m.media-amazon.com/images/G/35/gno/sprites/nav-sprite-global-2x-hm-dsk-reorg._CB405936783_.png'
```

In [7]:

```python
soup.prettify()
```

Out[7]:

```
'<!DOCTYPE html>\n<html class="a-no-js" data-19ax5a9jf="dingo" lang="en-au">\n <!-- sp:feature:head-start -->\n <head>\n  <scri
pt>\n   var aPageStart = (new Date()).getTime();\n  </script>\n  <meta charset="utf-8"/>\n  <!-- sp:end-feature:head-start -->\
n  <!-- sp:feature:csm:head-open-part1 -->\n  <!-- sp:end-feature:csm:head-open-part1 -->\n  <!-- sp:feature:cs-optimization --
>\n  <meta content="on" http-equiv="x-dns-prefetch-control"/>\n  <link href="https://images-fe.ssl-images-amazon.com" rel="dns-
prefetch"/>\n  <link href="https://m.media-amazon.com" rel="dns-prefetch"/>\n  <link href="https://completion.amazon.com" rel="
dns-prefetch"/>\n  <!-- sp:end-feature:cs-optimization -->\n  <!-- sp:feature:csm:head-open-part2 -->\n  <!-- sp:end-feature:cs
m:head-open-part2 -->\n  <!-- sp:feature:aui-assets -->\n  <link href="https://m.media-amazon.com/images/I/11EIQ5IGqaL._RC|01ZT
HTZObnL.css,410yLeQZHKL.css,31OSFXVtM5L.css,013z33uKh2L.css,017DsKjNQJL.css,0131vqwP5UL.css,41EWOOlBJ9L.css,11TIuySqr6L.css,01E
lnPiDxWL.css,11fJbvhE5HL.css,01Dm5eKVxwL.css,01IdKcBuAdL.css,01y-XAlI+2L.css,21P6CS3L9LL.css,01oDR3IULNL.css,41Js1DVdbVL.css,01
XPHJk60-L.css,01S0vRENeAL.css,21IbH+SoKSL.css,11MrAKjcAKL.css,21fecG8pUzL.css,11a5wZbuKrL.css,01CFUgsA-YL.css,31pHA2U5D9L.css,1
1qour3ND0L.css,116t+WD27UL.css,11gKCCKQV+L.css,11061HxnEvL.css,11oHt2HYxnL.css,01j2JE3j7aL.css,11JQtnL-6eL.css,21KA2rMsZML.css,
11jtXRmppwL.css,0114z6bAEoL.css,21uwtfqr5aL.css,11QyqG8yiqL.css,11K24eOJg4L.css,11F2+OBzLyL.css,01890+Vwk8L.css,01g+cOYAZgL.cs
s,01cbS3UK11L.css,21F85am0yFL.css,01giMEP+djL.css_.css?AUIClients/AmazonUI&amp;S5swH4CQ#not-trident.388250-T1.432724-T1.577951-
T1.577971-T1.577969-T1.632675-T1.577970-T1.577878-T1" rel="stylesheet">\n   <script>\n    (function(d,g,R,G){function v(a){w&&
w.tag&&w.tag(k(":","aui",a))}function n(a,b){w&&w.count&&w.count("aui:"+a,0===b?0:b||(w.count("aui:"+a)||0)+1)}function q(a){tr
y{return a.test(navigator.userAgent)}catch(b){return!1}}function r(a){return"function"===typeof a}function z(a,b,c){a.addEventL
istener?a.addEventListener(b,c,!1):a.attachEvent&&a.attachEvent("on"+b,c)}function k(a,b,c,d){b=b&&c?b+a+c:b||c;return d?k(a,b,
d):b}function H(a,b,c){try{Object.defineProperty(a,b,{value:c,writable:!1})}catch(u){a[b]=\nc}return c}function ua(a,b,c){var d
=c=a.length,f=function(){d--||(S.push(b),T||(setTimeout(ea,0),T=!0))};for(f();c--;)fa[a[c]]?f():(C[a[c]]=C[a[c]]||[]).push(f)}f
```

Show solution

> The first value of the list, returned by the `findAll` fuction is the `html` code containing the title of the first book. We note that the code is long and does not allow the title to be found easily.
>
> Indeed, the name of the tags is too general to find specific information. We would have to find a more precise way to identify an element.

2. Attributes

> An attribute is a localized instruction inside a tag. It provides additional characteristics of the tag. Not all tags have an attribute, but when this is the case, it is placed after the name of the opening tag. The name of the attribute is followed by an equal sign and a value placed in quotes:
>
> `<tag> attribute = 'value' >...</tag>`
>
> HTML elements may contain more than one attributes and are then separated by a space. Below some example of attributes:
>
> | Attribute | Role |
> |---|---|
> | `href` | Defines the link address |
> | `alt` | Defines a text relating to images (displayed if the image cannot be loaded) |
> | `src` | Indicates the source of the element |
> | `lang` | Indicates the language of the document |
> | `id` | Defines the unique ID of the element |
> | `class` | Indicates the name of the CSS class to use |
> | `style` | Defines the CSS style for the element |
>
> The attributes are essential for navigating properly in the HTML tree. They allow both to identify a tag with more precision and to store data that can be likely to interest us.

- **d)** Identify the tag and attribute containing information from the first book and those of the second book. What do you notice ?

> It is necessary to identify the HTML elements of the rectangle containing all the information of the first book. Then the elements of the rectangle of the second

book.

In [20]:

```
### Insert your code

soup.findAll('id')

list_book = soup.findAll('div', {'id' : 'gridItemRoot'})

for i in list_book[:3]:
    print(i.text)
```

```
#1RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cookNagi Maehashi4.9 out of 5 stars 1,118Paperback7 offers
from $35.99
#2Outlive: The Science and Art of LongevityMD, Peter Attia,4.7 out of 5 stars 622Paperback6 offers from $22.79
#3Atomic Habits: the life-changing million-copy #1 bestsellerJames Clear4.6 out of 5 stars 77,210Paperback10 offers from $24.24
```

Show solution

> In the HTML document of the Amazon page, we have noticed that, for different books, the same information is housed in the same types of beacons. In reality, this observation is not trivial. On the page, the books are represented in a similar way, it is therefore logical that the structure of the web code is the same.
>
> The `id` , `class` and `style` attributes are special. They are used to format the element to which they refer, and therefore host CSS language.
>
> To navigate using attributes, there are several ways:
>
> - Access the tag concerned thanks to the `find` method (or by tags), then recover the information by specifying between crochet the attribute which lodges it:
>
> ```
> soup.find('tag_name')['attribute_name']
> ```
>
> - Using the parameter `attrs` of the `findAll` method allowing to specify the name and value of an attribute:
>
> ```
> soup.findAll('tag_name', attrs = {'attribute_name' : 'attribute_value'})
> ```

- **e)** Thanks to the `findAll` function and thanks to the previous question, recover the data from all the books in a `bestsellers` variable.

> This step is to recover the source code relating only to books and not the source code of the whole page.

In [24]:

```
### Insert your code


list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

for i,j in enumerate(list_book[:3]):
    print(i+1,j.text)
```

```
1 RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cookNagi Maehashi4.9 out of 5 stars 1,118Paperback7 offers
from $35.99
2 Outlive: The Science and Art of LongevityMD, Peter Attia,4.7 out of 5 stars 622Paperback6 offers from $22.79
3 Atomic Habits: the life-changing million-copy #1 bestsellerJames Clear4.6 out of 5 stars 77,210Paperback10 offers from $24.24
```

Show solution

> Identifying the tag that contains information from all books is not necessarily successful the first time. You can check the length of the `bestsellers` variable: If the number obtained is equal to the number of books visible on the page, it is because you have found it. Otherwise we will have to try with the parents of the tag or the children.
>
> At first, we will recover the information from the first book by identifying their tags. Then in a second step, we will generalize our research for all the books on the page.

- **f)** Recover all the data from the first book on the page in a `bestseller` variable, then display it.
- **g)** By sailing in the sub-rib `bestseller` , recover the title, the writer, the number of votes and the price of the first book in the variables `title` , `writer` and `price` .

In [32]:

```
bestseller1 = bestsellers[0]
bestseller1.text
```

Out[32]:

```
'RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cookNagi Maehashi4.9 out of 5 stars\u20091,118Paperback7 off
ers from $35.99'
```

In [104]:

```
list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

for n,i in enumerate(list_book[:3]):
    title = i.find('img')['alt']
    writer = i.find('div', {'class' : 'a-row a-size-small'}).text
    price = i.find('span', {'class': "p13n-sc-price"}).text
    star = i.find('div', {'class' : 'a-icon-row'}).text
    link = i.find('a')['href']
    #print(star)
    #print(f'https://www.amazon.com.au/{link}')
    print(f'Title {n+1} : {title}\nWriter {n+1} : {writer}\nPrice {n+1} : {price}\nstar: {star}')
```

```
Title 1 : RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cook
Writer 1 : Nagi Maehashi
Price 1 : $35.99
star: 4.9 out of 5 stars 1,118
Title 2 : Outlive: The Science and Art of Longevity
Writer 2 : MD, Peter Attia,
Price 2 : $22.79
star: 4.7 out of 5 stars 622
Title 3 : Atomic Habits: the life-changing million-copy #1 bestseller
Writer 3 : James Clear
Price 3 : $24.24
star: 4.6 out of 5 stars 77,210
```

In [43]:

```
### Insert your code

list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

for n,i in enumerate(list_book[:3]):
    title = i.find('img')['alt']
    writer = i.find('div', {'class' : 'a-row a-size-small'}).text
    price = i.find('span', {'class': "p13n-sc-price"}).text
    print(f'Title {n+1} : {title}\nWriter {n+1} : {writer}\nPrice {n+1} : {price}')
```

```
Title 1 : RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cook
Writer 1 : Nagi Maehashi
Price 1 : $35.99
Title 2 : Outlive: The Science and Art of Longevity
Writer 2 : MD, Peter Attia,
Price 2 : $22.79
Title 3 : Atomic Habits: the life-changing million-copy #1 bestseller
Writer 3 : James Clear
Price 3 : $24.24
```

Show solution

> Sometimes certain elements are encoded. To recover clean data, you have to get rid of the superfluous characters. Several means can be used:
>
> - Text mining methods.
> - The `UnicodeDammit` function of the `BeautifulSoup` module identifies the language of the code and transforms it into text characters.
> - The `strip()` method allows you to delete any character informed in argument. If no argument is specified, all the spaces are deleted.

- **h)** Import `UnicodeDammit` from the `bs4` module and decode the necessary elements. The `strip()` method can also be used.

In [48]:

```
### Insert your code

from bs4 import UnicodeDammit

dammit_price = UnicodeDammit('4,99€')
dammit_price
dammit_price.unicode_markup
```

Out[48]: '4,99€'

Show solution

Retrieve data

> From the platform, you can not recover the data in an external file. It's why, in this part, we will only collect data in a Dataframe.

- **i)** Import `pandas` and recover all items for all objects in a `books` Dataframe.

> We can use a `for` loop to iterate on the bestsellers variable.

In [59]:

```
for i in list_book[:3]:
    title = i.find('img')['alt']
    writer = i.find('div', {'class' : 'a-row a-size-small'}).text
    price = i.find('span', {'class': "p13n-sc-price"}).text
    print(title, writer, price)
```

```
RecipeTin Eats: Dinner: 150 recipes from Australia's most popular cook Nagi Maehashi $35.99
Outlive: The Science and Art of Longevity MD, Peter Attia, $22.79
Atomic Habits: the life-changing million-copy #1 bestseller James Clear $24.24
```

In [1]:

```python
from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
import pandas as pd
import numpy as np


url = 'https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0'
page = urlopen(url)

soup = bs(page, "html.parser")

list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

list_title, list_writer, list_price, list_star, list_link = [], [], [], [], []
list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

for i in list_book:
    title = i.find('img')['alt']
    writer = i.find('div', {'class' : 'a-row a-size-small'}).text
    price = i.find('span', {'class': "p13n-sc-price"}).text
    try:
        star = i.find('div', {'class' : 'a-icon-row'}).text
    except:
        star = np.nan
    link = i.find('a')['href']
    list_title.append(title)
    list_writer.append(writer)
    list_price.append(price)
    list_star.append(str(star)[:3])
    list_link.append(f'https://www.amazon.com.au/{link}')

dict = {'title': list_title, 'writer':list_writer, 'price':list_price, 'star':list_star, 'link':list_link}

df = pd.DataFrame(dict)

print(df.shape)
df.head(10)
```

```
(50, 5)
```

Out[1]:

	title	writer	price	star	link
0	RecipeTin Eats: Dinner: 150 recipes from Austr...	Nagi Maehashi	$35.99	4.9	https://www.amazon.com.au//RecipeTin-Eats-reci...
1	Outlive: The Science and Art of Longevity	MD, Peter Attia,	$22.79	4.7	https://www.amazon.com.au//Outlive-Longevity-P...
2	Atomic Habits: the life-changing million-copy ...	James Clear	$24.24	4.6	https://www.amazon.com.au//Atomic-Habits-Prove...
3	Lessons in Chemistry: The No. 1 Sunday Times b...	Bonnie Garmus	$12.00	4.6	https://www.amazon.com.au//Lessons-Chemistry-S...
4	No More Nappies: A Potty-Training Book	Marion Cocklico	$16.69	4.7	https://www.amazon.com.au//No-More-Nappies-Pot...
5	The Ashes and the Star-Cursed King	Carissa Broadbent	$31.39	4.7	https://www.amazon.com.au//Ashes-Star-Cursed-K...
6	Twenty Thousand Fleas Under the Sea (Dog Man #11)	Dav Pilkey	$9.00	4.7	https://www.amazon.com.au//Twenty-Thousand-Fle...
7	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	$12.00	4.5	https://www.amazon.com.au//Seven-Husbands-Evel...
8	Ikigai: The Japanese secret to a long and happ...	Héctor García	$19.28	4.4	https://www.amazon.com.au//Ikigai-Japanese-sec...
9	The Barefoot Investor	Scott Pape	$19.00	4.8	https://www.amazon.com.au//Barefoot-Investor-S...

Show solution

It is important to repeat that since the source code of a web page can constantly change, it is necessary, once the data are recover, save them in an external file. Thus, apart from the Datascientest platform, you can recover the data in a CSV file on your local computer, thanks to the following commands:

```python
filename = 'file_name.csv'

f = open(filename, 'w')
f.write('column_1, column_2, column_3, column_4\n')
f.write(data_col_1 + ',' + data_col_2 + ',' + data_col_3 + ',' + data_col_4 + '\n')
f.close()
```

The information is saved on your computer and usable for your projects.

Conclusion

`BeautifulSoup` is a `Python` module which allows you to create and personalize a web scraper. The tool does not require a particular acuity in computer science, only a few notions in Python and Web programming are enough. The other `Python` modules have more features and therefore no longer have an expertise. In particular, `Selenium` is able to navigate the web and therefore automates data collection on multiple web pages.

Note that there are also pre-defined web scrapers. For example, Chrome extensions are widely used in digital marketing. They are applications that can be integrated into your browser to analyze the online data. Other tools like software such as `parsehub` are very popular because they are very easy to use and do not necessarily require development in development.

In [2]:

```python
# RECAP WEB SCRAPING

from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
import pandas as pd
import numpy as np


url = 'https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0'
page = urlopen(url)

soup = bs(page, "html.parser")

list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

list_title, list_writer, list_price, list_star, list_link = [], [], [], [], []
list_book = soup.findAll('div', {'class' : 'zg-grid-general-faceout'})

for i in list_book:
    title = i.find('img')['alt']
    writer = i.find('div', {'class' : 'a-row a-size-small'}).text
    price = i.find('span', {'class': "p13n-sc-price"}).text
    try:
        star = i.find('div', {'class' : 'a-icon-row'}).text
    except:
        star = np.nan
    link = i.find('a')['href']
    list_title.append(title)
    list_writer.append(writer)
    list_price.append(price)
    list_star.append(str(star)[:3])
    list_link.append(f'https://www.amazon.com.au/{link}')

# Create DataFrame 01
dict = {'title': list_title, 'writer':list_writer, 'price':list_price, 'star':list_star, 'link':list_link}
df = pd.DataFrame(dict)

# Create DataFrame 02
df = pd.DataFrame(list(zip(list_title, list_writer, list_price, list_star, list_link)),
                  columns=["title", "writer", "price", "star", "link"])


print(df.shape)
df.head(10)
```

```
(50, 5)
```

Out[2]:

	title	writer	price	star	link
0	RecipeTin Eats: Dinner: 150 recipes from Austr...	Nagi Maehashi	$35.99	4.9	https://www.amazon.com.au//RecipeTin-Eats-reci...
1	Outlive: The Science and Art of Longevity	MD, Peter Attia,	$22.79	4.7	https://www.amazon.com.au//Outlive-Longevity-P...
2	Atomic Habits: the life-changing million-copy ...	James Clear	$24.24	4.6	https://www.amazon.com.au//Atomic-Habits-Prove...
3	Lessons in Chemistry: The No. 1 Sunday Times b...	Bonnie Garmus	$12.00	4.6	https://www.amazon.com.au//Lessons-Chemistry-S...
4	No More Nappies: A Potty-Training Book	Marion Cocklico	$16.69	4.7	https://www.amazon.com.au//No-More-Nappies-Pot...
5	The Ashes and the Star-Cursed King	Carissa Broadbent	$31.39	4.7	https://www.amazon.com.au//Ashes-Star-Cursed-K...
6	Twenty Thousand Fleas Under the Sea (Dog Man #11)	Dav Pilkey	$9.00	4.7	https://www.amazon.com.au//Twenty-Thousand-Fle...
7	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	$12.00	4.5	https://www.amazon.com.au//Seven-Husbands-Evel...
8	Ikigai: The Japanese secret to a long and happ...	Héctor García	$19.28	4.4	https://www.amazon.com.au//Ikigai-Japanese-sec...
9	The Barefoot Investor	Scott Pape	$19.00	4.8	https://www.amazon.com.au//Barefoot-Investor-S...

In [ ]:

✖ Unvalidate