DataScientest • com

Web scraping with Python

Inspect a web page

Context

What is a web?

World Wide Web, or more simply the web is a tool for consulting websites via a browser (Firefox, Chrome, Safari, ...).
This web interacts with servers to share the hypertext documents they host. These files are identified by their URL which is an IP address, actually addressing the computer on which the site is hosted.
A web page consists of a hypertext document and various files. Thus, a website is a set of web pages linked to each other by links to easily navigate from one document to another.

What is the web scraping and what is its use?

Web scraping is a technique to automate data recovery from web pages. It collects the content of the sites and extract the relevant data.

In general, scraping is widely used in the marketing and insurance sector. Small as large companies use it to monitor and compare the prices of online offers, ensure their presence and reputation on the web and much more.

In the field of artificial intelligence (AI), scraping is an essential tool. The AI depends on the data and thus, automating the extraction of the data makes it possible to facilitate and optimize automatic learning. Therefore, this tool is widely used to create or complete databases for all types of projects.

Scraping challenges

At first, it is necessary to access the source code of the web pages that we want to scrap. However, this source code can constantly change and make scraping obsolete. It is therefore necessary, once the data has been recovered, save them in a file on a local computer.

Scraping is not necessarily an obvious practice. The two main obstacles that a scraper may meet are as follows.

- As we have just mentioned, data collection is carried out by accessing the source code of web pages, so any modification of the structure of a web page can make scraping obsolete. Thus, the updates of websites are real challenges at web scraping.
- In addition, some sites try to protect themselves from scrapers by implementing different strategies to make recovery of data tedious. Anti-scraping technologies or invisible traps (called HoneyPots) can thus slip into the web page and block the actions of the scraper.

Tool right

Collecting data using scraping web tools is completely legal. On the other hand, it is the use of the data later that can be problematic. Each site has its own conditions of use and it is important to document yourself on this subject before any illegal maneuver.

Goals

In this module, we will take an example a page of the Amazon site. This page is accessible via any browser thanks to the site URL (https://www.amazon.fr).
The objective will be to recover information concerning the best sales of literature books.
If the site is not accessible (for maintenance or any other reason), it is quite possible to practice on another web page.

More precisely, this module will allow you to:

- understand and know how to read the sources of a web page
- knowing how to navigate in an HTML document and identify page data
- recover site data

The content of the web pages can be coded in different programming languages. This content is herberged in one or more documents whose type varies depending on the language used. In most cases, a web page is coded, among other things, with HTML and CSS languages and is therefore made up of an HTML document and a CSS file.

Example of an HTML document (left) and a CSS file (right)

```html
<!DOCTYPE html>
<html lang="fr">

<head>
        <title> Introduction HTML </title>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="css.css" type="text/css" />
</head>

<body>
        <div id="entete_de_page">
                <p> Introduction HTML <br/>
                cours et exercices </p>
        </div>

        <div id="contenu">
                <h1> Cours </h1>
                <img src="html.jpg" alt="Image d'un arbre HTML" width="250"/>
                <h2> Html </h2>
                <p> est un langage de balises. <br/>
                Il est souvent utilisé pour créer une page Web.</p>
                <p> Souvent associé avec d'autres fichiers <br/> il permet de mettre en forme
une page Web. </p>
                <br/> <br/>

                <hr>

        </div> <br/>

        <div id="pied_de_page">
                <p> Les exercices sont <br/>
                pas difficile </p>
        </div> <br/>

</body>

</html>
```

```css
body, div {
        margin: 0;
        padding: 0;
}

body {
        background-color: lightblue;
        font-family: calibri;
        font-size: 18px;
}

img {
        float: right;
        margin-right: 60px;
}

th { width: 200px;}
ul { list-style-type: square; }

#entete_de_page {
        padding: 10px;
        height: 60px;
        background-color: lightgreen;
        font-size: 18px;
        font-weight: bold;
        text-align: center;
}

#contenu {
        padding: 10px;
        background-color: lightblue;
}

#pied-de-page {
        padding: 10px;
        background-color: lightgreen;
        font-size: 10px;
}
```

Using a right click with your mouse, it is possible to access the source web page code on which you are. The inspectory 'button or display the source code of the page` then allows you to display the different languages with which the page is coded.

In [1]:

```
### Insert your solution
import pandas as pd
```

Hide solution

In [ ]:

```
print("At the top, we have access to the Amazon HTML document. At the bottom, we observe the CSS code of the page.")
```

But before using the necessary tools to recover the source code and extract the data, it is important to know how to identify and understand the different web languages that you risk meeting during your navigations.

Introduction to web languages

Web pages, intended to be deployed on the internet, can be published in several languages (HTML, GML, XML etc). The main web languages used to build web applications are the following three:

1. HTML

HTML for HyperText Markup Language allows the management and organization of the content of the page. This language is hosted in an HTML document.

Image of the HTML code of the Amazon page with closed tags

```
<!DOCTYPE html> == $0
<html lang="fr-fr" class=" a-js a-audio a-video a-canvas a-svg a-drag-drop a-geolocation a-history a-webworker a-autofocus a-input-placeholder a-textarea-placeholder a-local-storage a-gradients a-hires a-transform3d a-touch-scrolling a-text-shadow a-text-stroke a-box-shadow a-border-radius a-border-image a-opacity a-transform a-transition a-ember" data-19ax5a9jf="dingo" data-aui-build-date="3.22.1-2022-05-06">
  <!-- sp:feature:head-start -->
  <head>…</head>
  <!-- sp:end-feature:head-close -->
  <!-- sp:feature:start-body -->
  <body class="a-m-fr a-aui_72554-c a-aui_accordion_a11y_role_354025-c a-aui_killswitch_csa_logger_372963-c a-aui_launch_2021_ally_fixes_392482-c a-aui_pci_risk_banner_210084-c a-aui_preload_261698-c a-aui_rel_noreferrer_noopener_309527-c a-aui_template_weblab_cache_333406-c a-aui_tnr_v2_180836-c a-meter-animate">…</body>
</html>
```

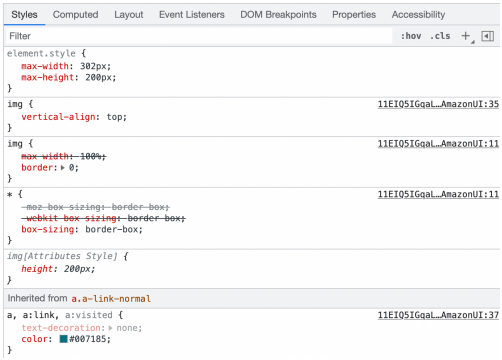
An HTML page is organized as a tree structure of subcategories structured by tags surrounded by characters `< >`. The minimum code to create a valid HTML page is made up of the following tags:

- `<!DOCTYPE html>` : an HTML document always starts by specifying the type of the document.
- `<html>` : the opening tag <html> and the closing tag </html> frame the entire page code. The document is then split in half, the head and the body.
- `<head>` : the tag <head> is a header element. It contains general characteristics of the document such as nature and content or the title of the page.
- `<title>` : allows you to indicate the title of the visible page on the top of the tabs of your browser.
- `<meta>` : this tag makes it possible to communicate to navigators the metadata on the page by giving the nature and content of the page.
- `<body>` : contains the elements defining the content of the page to the user as the different texts, images, etc.

2. CSS

CSS for Cascading Style Sheets, allows managing the appearance of the web page (colors, size, font, layout, etc.).
It completes the HTML and can be directly included in an HTML document by being stored in the tags `<style>` generally located in the header element `<head>` . Otherwise, it is hosted in a dedicated and stored CSS file in the element `<link/>` also located in `<head>` in most of the cases.

Image of the CSS code of the Amazon page



3. Javascript

In [2]:
```
### Insert your code
```

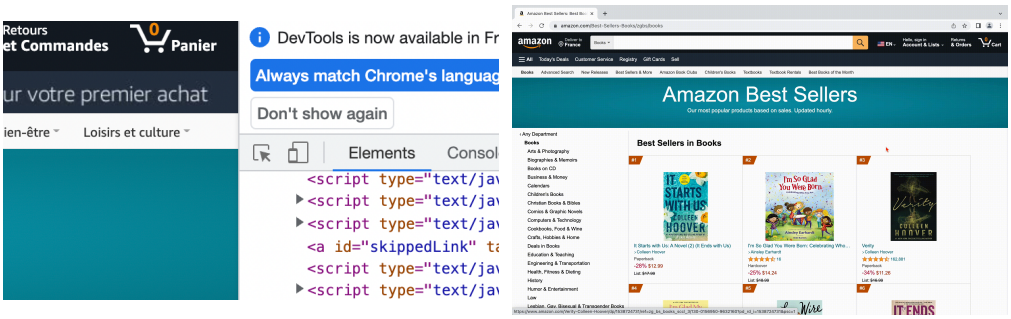
Hide solution

In [5]:
```
print("There are two ways to identify that it is indeed an interactive web page:\n"
"1. The page contains many interaction buttons (image, title, ...)\n"
"2. Many tags <script> are present in the HTML document.")
```

```
There are two ways to identify that it is indeed an interactive web page:
1. The page contains many interaction buttons (image, title, ...)
2. Many tags <script> are present in the HTML document.
```

After accessing the source code of the web page, you must be able to find the information visible on the web page. For this, you can:

- navigate with your mouse in the available code after inspected the page (each tag is associated with part of the page).
- click on the arrow visible on the image below and then identify the information by moving your mouse on the data. Three elements are displayed: the tag, the name of the class and the size as well as the location of the element code in the HTML document visible on the left.



- **e)** Display the information tags (image, title, prize, ...) of the first book.
- **f)** Locate the source code relating to the various information in the book. For each information, what word follows the name of the tag?

In [3]:
```
### Insert your code
```

Hide solution

In [4]:

```
### The solution can be obsolete due to the constant modifications of the Amazon source code

# By clicking on the arrow mentioned above, you can easily identify the tags (purple characters)

print("Image: img \n",
      "Titre: div \n",
      "Ecrivain: div \n",
      "Format: span \n",
      "Prix: span \n")

print('For each information, the tags are followed by the word class')
```

```
Image: img
 Titre: div
 Ecrivain: div
 Format: span
 Prix: span

For each information, the tags are followed by the word class
```

> The web scraping is carried out in several steps: recover the source code, find the information that interests us and extract it. \ Consequently, identifying the tag and, more generally, locating the code of a data is essential to recover the information of the site.

Main Python modules

> The objective being to collect information in an automated manner, we will then use a scraping web module which will allow this task to be completed.
> Python has several modules allowing the consultation of web resources, here are the three main ones:

1. BeautifulSoup

> Simple and quick to handle, Beatifulsoup is the ideal module to start in the web scraping. It is for this reason that it will be introduced to you in the rest of the module.
> On the other hand, this tool is not very robust and is therefore not the most suitable for carrying out complex projects.

2. Scrapy

> Faster and more complete, Scrapy is a more efficient module than BeautifulSoup. More difficult to master, it is a better choice for delicate projects.

3. Selenium

> Selenium is above all a tool that was mainly created for automated web tests. Due to its compatibility with JavaScript, it is also used for web scraping. And, unlike the two previous modules, since Selenium deciphers JavaScript, he is able to do dynamic scraping. It is a very efficient and fast tool that is more or less as efficient as Scrapy.

DataScientest • com

Web scraping with Python

Request and BeautifulSoup

In order to achieve the final goal of the web scraping, it is first necessary to recover the source code from a web page. In a second step, it is important to understand the structure of the recovered code to be able to navigate and collect the data we care about.

The objectives of this notebook are as follows:

- Create an object `BeautifulSoup`
- Navigate in the tree of an HTML document using the tags.

The notebook is based on the example of the Amazon page of best books available here (https://www.amazon.com.au/gp/bestsellers/?ref_=nav_cs_bestsellers).

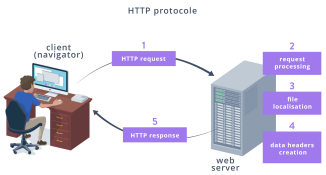
If the link does not work, it is possible to access the page by going to the site via your browser, or carry out the exercise from another page.

Introduction to Request

1. HTTP

The HTTP protocole (Hypertext Transfer Protocol) is a protocol used by the web to establish communication between customers and servers or, more simply, between the web browser of a person consulting a site and the machine hosting the site.
For the customer to access the site's resources, he uses the HTTP protocol. Depending on the type of request, it uses different methods asking the server to perform the requested action and display the requested resource.
Below is three examples of methods used by the protocol:

- `GET` : recover the content of a resource (using the URL)
- `PUT` : create or replace the content of a resource.
- `HEAD` : request only information on the resource, without asking for the resource itself.



This protocol is essential to access the content of an external resource. It is also identifiable in the URL link of web pages.

2. Request

The `Request` module is a Python module allowing easily to make the HTTP protocol.
As part of the Scraping Web, we must make a request to recover the source code of the page.
The `urllib` module of the `Request` module allows very easily to carry out this step using its `urlopen()` function. The content of the web page you want to scrap is then recoverable by the function by entering the link.

- **a)** Import the `urlopen` function of the `Request` module.
- **b)** Recover and store the HTML code of the web page in a variable named `page` .

In [17]:
```
import urllib.request
from bs4 import BeautifulSoup

amazon_url = "https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0"
page = urllib.request.urlopen(amazon_url)
print(page)
```

```
<http.client.HTTPResponse object at 0x7ff9a5054a30>
```

Hide solution

In [25]:
```
import urllib.request
from bs4 import BeautifulSoup

amazon_url = "https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0"
page = urllib.request.urlopen(amazon_url)
```

> `BeautifulSoup` is a tool that requires dependencies. Indeed, before any use of this bookstore, it is necessary to call on `Request` and go through the previous content recovery stage.

BeautifulSoup

> `BeautifulSoup` is a Python module allowing to extract HTML file data. It is compatible with the vast majority of browsers and allows you to automate web data collection.
>
> Once the source code has been obtained, the second step to write a web-scraper is to convert this code into a `BeautifulSoup` object to be able to exploit it.
> To decipher this source code there are several ways to create a Soup (the BeautifulSoup object).
> `BeautifulSoup` provides different synthetic analyzers called parsers which will allow the page to be transformed into an usable document. According to the parsers chosen (html.parser, lxml, lxml-xml, xml, html5lib), the module created, from the same document, different trees depending on the language chosen. Note, that all the languages of the various analyzers are quite similar, with the exception of HTML Parsers and XML Parsers.
>
> - `soup = BeautifulSoup(page, 'html.parser')`
> - `soup = BeautifulSoup(page, 'xml.parser')`
>
> The web scraping method with this tool is a so - called static method because it ignores JavaScript and only exploits the HTML and CSS code on the web page considered.

In [19]:
```
### Insert your solution

from bs4 import BeautifulSoup as bs

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

print(soup)
```

```
<!DOCTYPE html>
<html class="a-no-js" data-19ax5a9jf="dingo" lang="en-au"><!-- sp:feature:head-start -->
<head><script>var aPageStart = (new Date()).getTime();</script><meta charset="utf-8"/>
<!-- sp:end-feature:head-start -->
<!-- sp:feature:csm:head-open-part1 -->
<!-- sp:end-feature:csm:head-open-part1 -->
<!-- sp:feature:cs-optimization -->
<meta content="on" http-equiv="x-dns-prefetch-control"/>
<link href="https://images-fe.ssl-images-amazon.com" rel="dns-prefetch"/>
<link href="https://m.media-amazon.com" rel="dns-prefetch"/>
<link href="https://completion.amazon.com" rel="dns-prefetch"/>
<!-- sp:end-feature:cs-optimization -->
<!-- sp:feature:csm:head-open-part2 -->
<!-- sp:end-feature:csm:head-open-part2 -->
<!-- sp:feature:aui-assets -->
<link href="https://m.media-amazon.com/images/I/11EIQ5IGqaL._RC|01ZTHTZObnL.css,410yLeQZHKL.css,31OSFXVtM5L.css,013z33uKh2L.cs
s,017DsKjNQJL.css,0131vqwP5UL.css,41EWOOlBJ9L.css,11TIuySqr6L.css,01ElnPiDxWL.css,11fJbvhE5HL.css,01Dm5eKVxwL.css,01IdKcBuAdL.c
ss,01y-XAlI+2L.css,21P6CS3L9LL.css,01oDR3IULNL.css,41Js1DVdbVL.css,01XPHJk60-L.css,01S0vRENeAL.css,21IbH+SoKSL.css,11MrAKjcAKL.
css,21fecG8pUzL.css,11a5wZbuKrL.css,01CFUgsA-YL.css,31pHA2U5D9L.css,11qour3ND0L.css,116t+WD27UL.css,11gKCCKQV+L.css,11061HxnEv
```

Hide solution

In [26]:
```
from bs4 import BeautifulSoup as bs

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

> If you display the Soup object, you get the HTML tree on the web page. To improve the structure and to have better visibility of the tree, you can use the `prettify()` method.

- **e)** Display the parser and identify five tags.

In [23]:
```
### Insert your solution

print(soup)
```

Show solution

> The parser of a web page is, in practice, quite difficult to read. Thus, to understand the structure of the code, we will base ourselves on an example of a less complex HTML document.

HTML tree navigation

Tree structure

An HTML page is organized as a tree structure of subcategories structured by tags also called TAG or HTML elements.

Example of an HTML tree

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title> HTML Introduction </title>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="'style1.css" type="text/css" />
    </head>

    <body>
        <div id="header">
            <p> HTML Introduction <br/>
            courses and exercices </p>
        </div>

        <div id="content">
            <h1> Course </h1>
            <img src="html.jpg" alt="image of an HTML tree" width="250" />

            <h2> Html </h2>
            <p> is a tag language <br/>
            and is used to ceate web pages.
            </p>
            <hr>
        </div>

        <div id="footer">
            <p> The exercices are not difficult. </p>
        </div>
    </body>
</html>
```

Tags

The pairs tags: open, contain text, and close. The character `/` points out that it is a closure tag.

Examples of html pairs

Tag	Role
`<a> </a>`	Hyperlink
`<h1> </h1>, <h2>...</h6>`	Section titles from level 1 to 6
`<div> </div>`	Division of content, it serves as a container for shaping in CSS.
`<p> </p>`	Paragraph
`<ul> </ul>, <ol> </ol>`	List with unnumbered and numbered bullets
`<li> </li>`	Bullet list items
`<table> </table>`	Table
`<tr> </tr>`	Table row
`<td> </td>`	Table cell
`<tbody> </tbody>`	Table body (groups one or more tags `<tr>`)
`<script>...</script>`	JavaScript code

Orphan tags are tags that are used to insert an element in a specific location (for example an image, a link). It is not necessary to delimit the beginning and the end of this order, it is simply written as this: `<tag />` .

Examples of HTML orphan tags

Tag	Role
`<img/>`	Image
` `	Return to line in a paragraph mainly
`<link/>`	Introduce a link to a CSS or JS file

Navigation

The objective being to recover the data from the tree, we seek to access the different tags. To do this, we can navigate in the tree thanks to the name of the tags.
For example to have the Amazon page title tag, you can use the following code:

```
soup.title
```

In [33]:
```
### Insert your solution

print(soup.h1.text)
```

```
Best Sellers in Books
```

In [39]:
```
soup
```

Out[39]: https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0 (https://www.amazon.com.au/gp/bestsellers/books/ref=zg_bs_nav_0)

Hide solution

In [28]:
```
print(soup.h1.text)

soup.div.div.a # The tag is the 1st tag at the time of creating the course
```

```
Best Sellers in Books
```

Out[28]:
```
<a aria-label="Amazon.com.au" class="nav-logo-link nav-progressive-attribute" href="/ref=nav_logo" id="nav-logo-sprites">
<span class="nav-sprite nav-logo-base"></span>
<span class="nav-sprite nav-logo-ext nav-progressive-content" id="logo-ext"></span>
<span class="nav-logo-locale">.com.au</span>
</a>
```

> Any text within a tag is considered an element of the HTML document. To recover it, it is possible to use the texts `text` or `string` .

- **h)** Recover the title of the page.

In [32]:
```
### Insert your solution


soup.title.string
```

Out[32]:
```
'Amazon.com.au Best Sellers: The most popular items in Books'
```

Hide solution

In [29]:
```
soup.title.string
```

Out[29]:
```
'Amazon.com.au Best Sellers: The most popular items in Books'
```

> The HTML tree is built from a 'parent' and its 'children'. A tag that contains other tags is the parent of the latter. And on the other hand, the tags contained are the children of the parent element.
> A child is not necessarily a tag but any element that an element can contain.
>
> In the example of the tree above, the `string` "Introduction to HTML" is the child of `title` which has as parent `head` .
>
> To navigate in the tree of a parent to a child or to access the descendants of a parent (in other words to access all the children of a parents), it is possible to use the generators `parent` , `children` (or `contents`) and `descendants` . Except for `contents` , the information in these orders must be recovered from a list, otherwise it cannot be read.

- **i)** Access the parent from the HTML element of the page. What do you notice ?
- **j)** What is the number of children in the `BeautifulSoup` object ?

In [31]:
```
### Insert your solution

print(list(soup.html.parent))
print("The BeautifulSoup object has children")
print("The beautifulSoup object has ", len(list(soup.descendants)), "childrens")
```

```
['html', <html class="a-no-js" data-19ax5a9jf="dingo" lang="en-au"><!-- sp:feature:head-start -->
<head><script>var aPageStart = (new Date()).getTime();</script><meta charset="utf-8"/>
<!-- sp:end-feature:head-start -->
<!-- sp:feature:csm:head-open-part1 -->
<!-- sp:end-feature:csm:head-open-part1 -->
<!-- sp:feature:cs-optimization -->
<meta content="on" http-equiv="x-dns-prefetch-control"/>
<link href="https://images-fe.ssl-images-amazon.com" rel="dns-prefetch"/>
<link href="https://m.media-amazon.com" rel="dns-prefetch"/>
<link href="https://completion.amazon.com" rel="dns-prefetch"/>
<!-- sp:end-feature:cs-optimization -->
<!-- sp:feature:csm:head-open-part2 -->
<!-- sp:end-feature:csm:head-open-part2 -->
<!-- sp:feature:aui-assets -->
<link href="https://m.media-amazon.com/images/I/11EIQ5IGqaL._RC|01ZTHTZObnL.css,410yLeQZHKL.css,31OSFXVtM5L.css,013z33uKh2L.cs
s,017DsKjNQJL.css,0131vqwP5UL.css,41EWOOlBJ9L.css,11TIuySqr6L.css,01ElnPiDxWL.css,11fJbvhE5HL.css,01Dm5eKVxwL.css,01IdKcBuAdL.c
ss,01y-XAlI+2L.css,21P6CS3L9LL.css,01oDR3IULNL.css,41Js1DVdbVL.css,01XPHJk60-L.css,01S0vRENeAL.css,21IbH+SoKSL.css,11MrAKjcAKL.
css,21fecG8pUzL.css,11a5wZbuKrL.css,01CFUgsA-YL.css,31pHA2U5D9L.css,11qour3ND0L.css,116t+WD27UL.css,11gKCCKQV+L.css,11061HxnEv
L.css,11oHt2HYxnL.css,01j2JE3j7aL.css,11JQtnL-6eL.css,21KA2rMsZML.css,11jtXRmppwL.css,0114z6bAEoL.css,21uwtfqr5aL.css,11QyqG8yi
```

Hide solution

In [30]:

```
print(list(soup.html.parent))
print("The BeautifulSoup object has children")
print("The beautifulSoup object has ", len(list(soup.descendants)), "childrens")
```

```
['html', <html class="a-no-js" data-19ax5a9jf="dingo" lang="en-au"><!-- sp:feature:head-start -->
<head><script>var aPageStart = (new Date()).getTime();</script><meta charset="utf-8"/>
<!-- sp:end-feature:head-start -->
<!-- sp:feature:csm:head-open-part1 -->
<!-- sp:end-feature:csm:head-open-part1 -->
<!-- sp:feature:cs-optimization -->
<meta content="on" http-equiv="x-dns-prefetch-control"/>
<link href="https://images-fe.ssl-images-amazon.com" rel="dns-prefetch"/>
<link href="https://m.media-amazon.com" rel="dns-prefetch"/>
<link href="https://completion.amazon.com" rel="dns-prefetch"/>
<!-- sp:end-feature:cs-optimization -->
<!-- sp:feature:csm:head-open-part2 -->
<!-- sp:end-feature:csm:head-open-part2 -->
<!-- sp:feature:aui-assets -->
<link href="https://m.media-amazon.com/images/I/11EIQ5IGqaL._RC|01ZTHTZObnL.css,410yLeQZHKL.css,31OSFXVtM5L.css,013z33uKh2L.cs
s,017DsKjNQJL.css,0131vqwP5UL.css,41EWOOlBJ9L.css,11TIuySqr6L.css,01ElnPiDxWL.css,11fJbvhE5HL.css,01Dm5eKVxwL.css,01IdKcBuAdL.c
ss,01y-XAlI+2L.css,21P6CS3L9LL.css,01oDR3IULNL.css,41Js1DVdbVL.css,01XPHJk60-L.css,01S0vRENeAL.css,21IbH+SoKSL.css,11MrAKjcAKL.
css,21fecG8pUzL.css,11a5wZbuKrL.css,01CFUgsA-YL.css,31pHA2U5D9L.css,11qour3ND0L.css,116t+WD27UL.css,11gKCCKQV+L.css,11061HxnEv
L.css,11oHt2HYxnL.css,01j2JE3j7aL.css,11JQtnL-6eL.css,21KA2rMsZML.css,11jtXRmppwL.css,0114z6bAEoL.css,21uwtfqr5aL.css,11QyqG8yi
```

> Now we can navigate the whole tree. However, remember that we are only interested in some specific information on the page and not all the tags, we will see in the next notebook how to access certain elements without browsing the entire HTML structure.

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]:

```
### 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_ROLE_354025":"T1","AUI_PRELOAD_261698":"C","AUI_TEMPLATE_WEBLAB_CACHE_333406":"C","AUI_72554":"C","AUI_KILLSWITCH_CSA_LOGGER_372963":"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.createElement("header");var r=function(n){function u(n,r,u){n[u]=function(){a._replay.push(r.concat(e(u,arguments)))}}var a={};return 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,[],"publish"),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]:

```
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  <script>\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:csm:head-open-part2 -->\n  <!-- sp:feature:aui-assets -->\n  <link href="https://m.media-amazon.com/images/I/11EIQ5IGqaL._RC|01ZTHTZObnL.css,410yLeQZHKL.css,31OSFXVtM5L.css,013z33uKh2L.css,017DsKjNQJL.css,0131vqwP5UL.css,41EWOOlBJ9L.css,11TIuySqr6L.css,01ElnPiDxWL.css,11fJbvhE5HL.css,01Dm5eKVxwL.css,01IdKcBuAdL.css,01y-XAlI+2L.css,21P6CS3L9LL.css,01oDR3IULNL.css,41Js1DVdbVL.css,01XPHJk60-L.css,01S0vRENeAL.css,21IbH+SoKSL.css,11MrAKjcAKL.css,21fecG8pUzL.css,11a5wZbuKrL.css,01CFUgsA-YL.css,31pHA2U5D9L.css,11qour3ND0L.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.css,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){try{return a.test(navigator.userAgent)}catch(b){return!1}}function r(a){return"function"===typeof a}function z(a,b,c){a.addEventListener?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
```

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]:

```
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:

```
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]:

```
# 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 [ ]: