Importation des libraries

```python
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
import pandas as pd
```

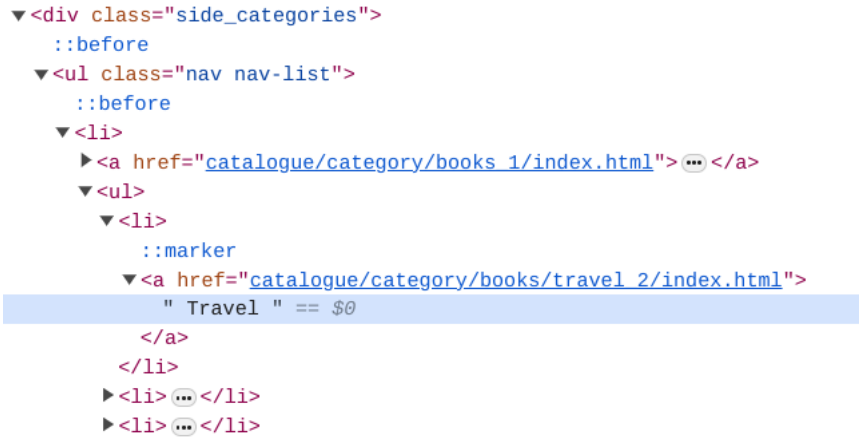
On récupère le code HTML de l'url

```python
url = "https://books.toscrape.com/index.html"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
```

On cherche toutes les catégories qu'on stock dans une liste.

```python
categories = soup.select('ul.nav-list > li > ul > li > a')
categories_list = [category.text.strip() for category in categories]
```

Cette méthode spécifie la hiérarchie d'éléments à rechercher. En décomposant cela :



— ul.nav-list : Recherche un élément $< ul >$ ayant la classe nav-list.
— $>$ li : Sélectionne un élément $< li >$ qui est après l'élément $< ul >$.
— $>$ ul : Sélectionne un élément $< ul >$ qui est après l'élément $< li >$.
— $>$ li : Sélectionne un élément $< li >$ qui est après l'élément $< ul >$.
— $>$ a : Sélectionne un élément $< a >$ qui est après l'élément $< li >$.

On crée une boucle qui permet de demander à l'utilisateur de saisir le nom de catégorie qu'il souhaite scrapper. Il a cinq catégories à choisir, donc la boucle s'exécute cinq fois. Si une catégorie n'existe pas, soit n'est pas présente dans la liste categories_list, alors un message d'erreur s'affiche pour entrer une nouvelle catégorie présente dans la liste.

```python
category_select = []

for i in range(5):
    while True:
        user_input = input(f"Entrez le nom de la catégorie {i + 1} à extraire : ")
        if user_input in categories_list and user_input not in category_select:
            category_select.append(user_input)
            break
```

```
        else:
            print("La catégorie n'est pas valide. Veuillez réessayer.")
```

Dans la variable 'category_select', nous devons obtenir une liste avec cinq catégories.

Désormais, on crée une liste 'book_data' qui va stocker les éléments qu'on va scrapper par livre. Pour chaque nom de catégorie, dans la variable 'category_select', on va parcourir le code HTML du lien associé à la catégorie. Cependant, les différentes catégories peuvent avoir plusieurs pages, alors on crée une condition pour exécuter le code tant qu'une nouvelle page existe.

Le code de la condition qui parcourt les pages, ainsi que la boucle qui parcourt les liens de chaque catégorie, est de la forme suivante :

```
book_data = []

for category_name in category_select:
    index = categories_list.index(category_name)
    category_link = urljoin(url, categories[index]['href'])

    page_number = 1
    while True:
        if page_number == 1:
            category_website = requests.get(category_link)
        else:
            category_website = requests.get(urljoin(category_link,
                                f"page-{page_number}.html"))


        soup = BeautifulSoup(category_website.text, 'html.parser')
        product_containers = soup.find_all('article', class_='product_pod')

        # Code qui récupérer des informations dans le code HTML de chaque page

            page_number += 1
        else:
            break  # la boucle s'arrête quand on atteint le nombre de page possible
```

Il faut parcourir l'ensemble du code HTML de chaque catégorie par page, pour obtenir le nom de chaque livre ainsi que son lien. La liste 'product_containers' stock le code HTML où se trouve chaque livre. Pour un seul livre, nous avons les informations suivantes (des informations sur l'image, nombre d'étoiles, le titre avec le lien du livre, le prix et le stock :

A la suite de cela, on effectue une boucle sur cette liste pour parcourir tous les livres qui se trouve sur le lien de la catégorie. Sur chaque livre, on récupère le titre, la présence d'un stock disponible, le nombre d'étoiles, lien de l'image et le lien de la page du livre de la façon suivante.

```
# Récupérer des informations dans le code HTML de chaque page
for container in product_containers:

# Titre du livre
title = container.find('h3').find('a')['title']

# Stocks
stock_available = container.find('p', class_='instock availability').text.replace('\n', ''

# Lien du livre
book_url = container.find('h3').find('a')['href']
book_url = f'https://books.toscrape.com/catalogue/{book_url}'
book_url = book_url.replace('../','')

# Lien de l'image
img_url = container.find('img')['src']
img_url = f'https://books.toscrape.com{img_url}'
img_url = img_url.replace('../','')

# Nombre d'étoiles
star_rating_class = container.find('p', class_='star-rating')['class'][1]
star_rating = {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4,
               'Five': 5}.get(star_rating_class, 0)
```

Le nombre d'étoiles est contenu dans la 'class'. Exemple : 'star-rating two'. Pour cela, on récupère le dernier mot de la class puis on le convertit en chiffre.
De plus, dans le code HTML, on constate que dans le lien de l'image et du livre, il y a des charactère type '../' en trop pour écrit qu'il soit écrit en entier. Il faut donc les supprimer afin d'obtenir un lien qui fonctionne, en le combinant avec le début du lien de la page d'accueil.

Il est possible d'obtenir plus d'informations en accédant à la page du libre. Pour cela, on récupère le code HTML du lien du livre préalablement enregistré dans la variable 'book_url'. Puis on parcoure le code HTML, pour obtenir des informations sur l'UPC, la description, le type de produit, le prix

avec la taxe et sans, la valeur de la taxe, le nombre de livres disponibles et le nombre de revues.

La structure du code HTML se présente comme suit :

```
<p class="instock availability"> == $0
  <i class="icon-ok">...</i>
   " In stock (19 available) "
  </p>
 <p class="star-rating Two">...</p>
  <hr>
 <div class="alert alert-warning" role="alert">...</div>
</div>
<!-- /col-sm-6 -->
::after
</div>
<!-- /row -->
<div id="product_description" class="sub-header">
  <h2>Product Description</h2>
</div>
<p>
  ""Wherever you go, whatever you do, just . . . don't do anything stupid." —My MotherDuring her
  yearlong adventure backpacking from South Africa to Singapore, S. Bedford definitely did a few
  things her mother might classify as "stupid." She swam with great white sharks in South Africa,
  ran from lions in Zimbabwe, climbed a Himalayan mountain without training in Nepal, and wa
  "Wherever you go, whatever you do, just . . . don't do anything stupid." —My MotherDuring her
```

Nous pouvons obtenir ici le nombre de libres en stocks et la description. Pour le nombre de livres en stock, on récupère le texte de la class "instock availability" et on remplace ' In stocks (' et ')' par rien.

```
<div class="sub-header">...</div> == $0
<table class="table table-striped">
 <tbody>
  <tr>
   <th>UPC</th>
   <td>a22124811bfa8350</td>
  </tr>
  <tr>
   <th>Product Type</th>
   <td>Books</td>
  </tr>
  <tr>
   <th>Price (excl. tax)</th>
   <td>£45.17</td>
  </tr>
  <tr>
   <th>Price (incl. tax)</th>
   <td>£45.17</td>
  </tr>
  <tr>
   <th>Tax</th>
   <td>£0.00</td>
  </tr>
  <tr>
   <th>Availability</th>
   <td>In stock (19 available)</td>
  </tr>
  <tr>
   <th>Number of reviews</th>
   <td>0</td>
  </tr>
 </tbody>
</table>
```

Pour récupérer l'UPC, le type de produit, les différents prix et le nombre de revues, nous devons parcourir les différents 'th' qui contient l'information que l'on souhaite extraire. Une fois le bon 'th' trouvé, on récupère le texte qui se trouve dans la partie 'td' juste après le 'th'.

Pour obternir ses informations, le code généré sur python est :

```
# Accéder à la page du détail du livre
book_detail_page = requests.get(book_url)
detail_soup = BeautifulSoup(book_detail_page.text, 'html.parser')

# Catégorie du livre
category = category_name

# Description
description = detail_soup.find('meta', {'name': 'description'})['content']
description = description.replace('\n    ', '').replace('\n', '')

# UPC
upc = detail_soup.find('th', string='UPC')
upc = upc.find_next('td').get_text(strip=True)

# Type de produit
product_type = detail_soup.find('th', string='Product Type')
product_type = product_type.find_next('td').get_text(strip=True)

# Price sans tax
price_excl_tax = detail_soup.find('th', string='Price (excl. tax)')
price_excl_tax = price_excl_tax.find_next('td').get_text(strip=True).replace('Â£', '')

# Prix avec la taxe
price_incl_tax = detail_soup.find('th', string='Price (incl. tax)')
price_incl_tax = price_incl_tax.find_next('td').get_text(strip=True).replace('Â£', '')

# Prix de la taxe
tax = detail_soup.find('th', string='Tax')
tax = tax.find_next('td').get_text(strip=True).replace('Â£', '')

# Nombre de livres disponibles
number_available = detail_soup.find_all('p', class_='instock availability')
number_available = number_available[0].get_text(strip=True).replace("In stock (", "").repl

# Nombre de revues
number_of_reviews = detail_soup.find('th', string='Number of reviews')
number_of_reviews = number_of_reviews.find_next('td').get_text(strip=True)
```

Chaque variable créée récupère les informations stockées dans celle-ci. Auparavant, nous avions créer une liste 'book_data' qui doit récupérer toutes ces informations. Nous pouvons désormais attribuer toutes ces variables à notre liste 'book_data' de la façon suivante.

```
# Ajouter les données du livre à la liste
book_data.append({
    'Title': title,
    'Book Link': book_url,
    'Image Link': img_url,
    'Description': description,
    'UPC': upc,
    'Category' : category,
```

```
    'Product Type': product_type,
    'Price £ (excl. tax)': price_excl_tax,
    'Price £ (incl. tax)': price_incl_tax,
    'Tax': tax,
    'Stock available' : stock_available,
    'Number of available': number_available,
    'Number of reviews': number_of_reviews
})
```

Chaque fois que la boucle s'exécute, cela créer un nouvel élément à la liste. Un élément, soit un livre, est représenté de la façon suivante dans la liste :

```
{'Title': 'Private Paris (Private #10)',
 'Book Link': 'https://books.toscrape.com/catalogue/private-paris-private-10_958/index.html',
 'Image Link': 'https://books.toscrape.commedia/cache/9d/05/9d0533bae1578846d728a82913b95c26.jpg',
 'Description': "Paris is burning--and only Private's Jack Morgan can put out the fire.When Jack Morgan stops by Private's Paris office, he e
nvisions a quick hello during an otherwise relaxing trip filled with fine food and sightseeing. But Jack is quickly pressed into duty after a
call from his client Sherman Wilkerson, asking Jack to track down his young granddaughter who is on the run f Paris is burning--and only Priva
te's Jack Morgan can put out the fire.When Jack Morgan stops by Private's Paris office, he envisions a quick hello during an otherwise relaxin
g trip filled with fine food and sightseeing. But Jack is quickly pressed into duty after a call from his client Sherman Wilkerson, asking Jac
k to track down his young granddaughter who is on the run from a brutal drug dealer.Before Jack can locate her, several members of France's cu
ltural elite are found dead--murdered in stunning, symbolic fashion. The only link between the crimes is a mysterious graffiti tag. As religio
us and ethnic tensions simmer in the City of Lights, only Jack and his Private team can connect the dots before the smoldering powder keg expl
odes. ...more",
 'UPC': 'b12b89017878a60d',
 'Category': 'Fiction',
 'Product Type': 'Books',
 'Price £ (excl. tax)': '47.61',
 'Price £ (incl. tax)': '47.61',
 'Tax': '0.00',
 'Stock available': '           In stock    ',
 'Number of available': '17',
 'Number of reviews': '0'},
```

Nous pouvons désormais convertir la liste 'book_data' en dataframe de type 'csv' de la façon suivante :

```
# Création du dataframe avec les 5 catégories
data_category = pd.DataFrame(book_data)
data_category.to_csv('books_data_category.csv', index = False)
```