**Extraire des données du web avec Python et certains de ses packages**

<https://openclassrooms.com/fr/courses/7168871-apprenez-les-bases-du-langage-python/exercises/4173>

**Description**

*Vous êtes chargé(e) de collecter des données de produits pour un commerçant en ligne. Utilisez Beautiful Soup pour extraire les informations des produits à partir de la page web du commerçant en ligne. Les données doivent être collectées pour toutes les pages de produits disponibles sur le site. Ensuite, exportez les données dans un fichier CSV ou JSON.*

*Pour ce quiz, utilisez cette URL fictive,   https://quiz-partie3.oc   , contenant le code HTML suivant :*

<!DOCTYPE html>

<html>

<head>

<title>E-commerce</title>

</head>

<body>

<h1>Bienvenue sur notre site web</h1>

<p>Nous sommes une entreprise qui propose une large gamme de produits dans différents domaines.</p>

<h2>Produits informatiques</h2>

<table>

<tr>

<th>Nom du produit</th>

<th>Description</th>

<th>Prix</th>

<th>Quantité en stock</th>

</tr>

<tr>

<td>Ordinateur portable</td>

<td>Processeur i7, 8Go RAM, SSD 256Go</td>

<td>1200$</td>

<td>10</td>

</tr>

<tr>

<td>Imprimante multifonction</td>

<td>Impression, numérisation, copie, fax</td>

<td>200$</td>

<td>5</td>

</tr>

<tr>

<td>Clé USB</td>

<td>32Go, USB 3.0</td>

<td>20$</td>

<td>50</td>

</tr>

</table>

<h2>Produits de beauté</h2>

<table>

<tr>

<th>Nom du produit</th>

<th>Description</th>

<th>Prix</th>

<th>Quantité en stock</th>

</tr>

<tr>

<td>Crème hydratante</td>

<td>Pour peaux sèches, 50ml</td>

<td>25$</td>

<td>20</td>

</tr>

<tr>

<td>Sérum anti-âge</td>

<td>Réduit les rides, 30ml</td>

<td>50$</td>

<td>10</td>

</tr>

<tr>

<td>Vernis à ongles</td>

<td>Couleur rouge, longue tenue</td>

<td>5$</td>

<td>100</td>

</tr>

</table>

</body>

</html>

Haut du formulaire

**Question 1**

Avant de commencer, vous devez installer le package Beautiful Soup à l'aide du gestionnaire de paquets pip pour analyser les données de la page HTML.

Quelle commande permet d’installer le package Beautiful Soup ?

pip install bs

pip install beautifulsoup4

install beautifulsoup4

pip beautifulsoup4

*Pour installer un nouveau package, on peut utiliser le gestionnaire de packages Python appelé pip  . Il suffit d'ouvrir un terminal ou une invite de commande, et d'exécuter la commande suivante :  pip install nom\_du\_package   . Remplacez  nom\_du\_package  par le nom du package que vous souhaitez installer. En vérifiant sur le registre public [PyPI](https://pypi.org/project/beautifulsoup4/), nous pouvons trouver que le nom du package est beautifulsoup4  .*

Bas du formulaire

**Question 2**

Correcte

Considérez que le package requests  est déjà installé dans votre environnement. Vous devez maintenant extraire le contenu de la page web du commerçant, et récupérer le contenu HTML de la page.  
Quelles lignes de code permettent d’extraire les données de la page du commerçant ? (Pour cette question, utilisez cette url fictive : https://quiz-partie3.oc.)

url = "https://quiz-partie3.oc"

response = requests.extract(url)

html = response.html

url = "https://quiz-partie3.oc"

response = requests.extract(url)

html = response.content

X

url = "https://quiz-partie3.oc"

response = requests.get(url)

html = response.content

url = "https://quiz-partie3.oc"

response = requests.get(url)

html = response.html.content

*On utilise la bibliothèque requests  pour envoyer une requête à l'URL que nous avons définie afin de récupérer les données HTML. La méthode get()  est utilisée pour obtenir la réponse du serveur. Une fois que nous avons récupéré la réponse du serveur, nous stockons le contenu HTML dans la variable html  . La fonction content  de la réponse nous donne accès au contenu brut de la page web.*

**Question 3**

Correcte

Une fois l’extraction terminée, vous pouvez enfin analyser le code HTML à l’aide du package Beautiful Soup.

Quelle ligne de code permet d’analyser le contenu HTML avec le package Beautiful Soup, et de le stocker dans une variable soup  ?

soup = BeautifulSoup(html, "html.parser")

BeautifulSoup(html, "html.parser")

soup = BeautifulSoup(html, "html")

soup = Beautiful\_Soup(html, "html.parser")

*La ligne de commande crée une instance de la classe BeautifulSoup  du package Beautiful Soup, en utilisant la variable html  qui contient le code HTML extrait à partir de la page web.*

*La classe BeautifulSoup  permet de parcourir et de manipuler le contenu HTML de manière plus facile et efficace que les méthodes de base de Python. Elle prend en argument le code HTML à parser et le type de parser à utiliser. Dans cet exemple, le type de parser utilisé est html.parser  .*

**Question 4**

Correcte

Avant de procéder à l'extraction de toutes les données, vous allez tout d’abord récupérer le titre de la page, car cela permettra de nommer le fichier CSV qui contiendra les données collectées.

Quelle ligne de code permet de récupérer le titre de la page HTML sous la forme d’une chaîne de caractères ?

soup.title

soup.title.string

soup.title.str

soup.html.title

La ligne soup.title.string  permet d'extraire le contenu de la balise title d'un document HTML analysé avec Beautiful Soup.

En détail,  soup  est l'objet BeautifulSoup qui représente l'arbre HTML parsé, et title  correspond à la balise  <title>  de la page HTML. En appelant .string  sur cette balise, on récupère la chaîne de caractères correspondant au contenu de la balise.

Ainsi, si on applique soup.title.string  sur le code HTML fourni, on obtiendra la chaîne de caractères "E-commerce"  , qui est le contenu de la balise <title>  .

**Question 5**

Correcte

Il est temps de passer aux choses sérieuses. Vous devez maintenant récupérer l’ensemble des informations de chaque produit, et les stocker dans une liste de dictionnaires. Chaque dictionnaire contiendra l’ensemble des informations d’un produit.

Quelles lignes de code correspondent à l’extraction des informations de l’ensemble des produits et à la sauvegarde dans une liste de dictionnaires ?

for table in soup.find\_all('table'):

for tr in table.find\_all('tr')[1:]:

td = tr.find\_all('td')

nom = td[0].get\_text()

description = td[1].get\_text()

prix = td[2].get\_text()

quantite = td[3].get\_text()

produit = {

'nom': nom,

'description': description,

'prix': prix,

'quantite': quantite

}

produits = []

for table in soup.find\_all('table'):

for tr in table.find\_all('tr'):

td = tr.find\_all('td')

nom = td[0].get\_text()

description = td[1].get\_text()

prix = td[2].get\_text()

quantite = td[3].get\_text()

produit = {

'nom': nom,

'description': description,

'prix': prix,

'quantite': quantite

}

produits.append(produit)

produits = []

for table in soup.find\_all('table'):

for tr in table.find\_all('tr')[1:]:

td = tr.find\_all('td')

nom = td[0]

description = td[1]

prix = td[2]

quantite = td[3]

produit = {

'nom': nom,

'description': description,

'prix': prix,

'quantite': quantite

}

produits.append(produit)

produits = []

for table in soup.find\_all('table'):

for tr in table.find\_all('tr')[1:]:

td = tr.find\_all('td')

nom = td[0].get\_text()

description = td[1].get\_text()

prix = td[2].get\_text()

quantite = td[3].get\_text()

produit = {

'nom': nom,

'description': description,

'prix': prix,

'quantite': quantite

}

produits.append(produit)

*La première boucle for  doit parcourir chaque table présente dans le document HTML en utilisant la méthode find\_all()  de BeautifulSoup, pour trouver toutes les balises <table>  .*

*La deuxième boucle for doit parcourir chaque ligne <tr>  de la table en utilisant la méthode find\_all()  de BeautifulSoup, pour trouver toutes les balises <tr>  , et en excluant la première ligne qui contient les en-têtes du tableau avec [1:] (Nom du produit, Description, Prix, Quantité en stock).*

*Les informations de chaque produit doivent être extraites en utilisant la méthode get\_text()  de BeautifulSoup sur les balises <td>  correspondantes, sinon vous allez avoir du code HTML. Les informations extraites peuvent ensuite être stockées dans un dictionnaire, et pour terminer, être ajoutées dans la liste avec la méthode append()  .*

**Question 6**

Correcte

Maintenant, vous devez passer à la phase de transformation, car tous les prix sur le site sont en dollars. Vous devez les convertir en euros en utilisant la formule suivante : prix\_euro = 0.8 \* prix\_dollar   . Assurez-vous d'arrondir les résultats à un maximum de deux chiffres après la virgule.

Quel bloc de code permet de transformer en euros la valeur en dollars ?

taux\_de\_change = 0.8

for produit in produits:

prix\_euro = produit['prix'].replace('€', '')

prix\_euro = float(prix\_euro)

prix\_dollar = prix\_euro \* taux\_de\_change

produit['prix'] = str(round(prix\_dollar, 2)) + '$'

taux\_de\_change = 0.8

for produit in produits:

prix\_euro = produit['prix'].replace('€', '')

prix\_euro =prix\_euro

prix\_dollar = prix\_euro \* taux\_de\_change

produit['prix'] = str(round(prix\_dollar, 2)) + '$'

taux\_de\_change = 0.8

for produit in produits:

prix\_euro = produit['prix'].replace('€', '')

prix\_euro = float(prix\_euro)

prix\_dollar = prix\_euro \* taux\_de\_change

produit['prix'] = str(prix\_dollar, 2) + '$'

taux\_de\_change = 0.8

for produit in produits:

prix\_euro = float(prix\_euro)

prix\_dollar = prix\_euro \* taux\_de\_change

produit['prix'] = str(round(prix\_dollar, 2))

*Il faut tout d’abord parcourir la liste produits qui contient des dictionnaires avec les informations sur chaque produit. Il faut extraire le prix en dollars, en n’oubliant pas de retirer le symbole dollar $  , et en le convertissant en float pour éviter une erreur lors du lancement du code. Une fois qu'il est converti en float, vous pouvez le multiplier par le taux de change. Enfin, il ne reste plus qu'à arrondir le résultat à 2 chiffres après la virgule, et à ajouter le signe  '€'  à la fin. Finalement, il faut mettre à jour la valeur de la clé 'prix' dans le dictionnaire pour chaque produit, avec le nouveau prix converti en euros.*

**Question 7**

Correcte

Vous pouvez maintenant vous attaquer à la dernière partie de l’ETL, le chargement des données dans un fichier CSV. Pour commencer, vous devez ouvrir un fichier CSV avec le bloc with  , qui contiendra ensuite le code permettant de remplir le fichier CSV. Je vous rappelle que le fichier doit avoir le nom du titre de la page extraite.

Quelle ligne de code permet d’ouvrir correctement le fichier pour le chargement des données ? Le titre de la page HTML se trouve dans la variable title  .

with open('produits.csv', mode='w', newline='') as file:

with open(f'{title}.csv', mode='w', newline='') as file:

with open(f'{'produits}.csv', mode='w', newline='') as file:

with open('produits.csv', mode='r', newline='') as file:

*L’instruction with  doit ouvrir le fichier en utilisant le mode d'écriture ( w   pour write), avec une variable nommée file  . Il ne faut pas oublier que le nom du fichier doit être composé du titre de la page HTML contenu dans la variable title  . Le paramètre newline  est utilisé pour spécifier le caractère de fin de ligne.*

**Question 8**

Correcte

Dernière ligne droite, vous devez maintenant insérer toutes les données dans le fichier en utilisant l’objet DictWriter()  . Effectivement, c’est le plus efficace, car notre liste comprend un ensemble de dictionnaires.

Quel bloc de code faut-il placer dans le bloc with  pour insérer les données dans le fichier CSV ?

fieldnames = ['nom', 'description', 'prix', 'quantite']

writer = csv.Writer(file, fieldnames=fieldnames)

writer.writeheader()

for produit in produits:

writer.writerow(produit)

fieldnames = ['nom', 'description', 'prix', 'quantite']

writer = csv.DictWriter(file, fieldnames=fieldnames)

for produit in produits:

writer.writerow(produit)

fieldnames = ['nom', 'description', 'prix', 'quantite']

writer = csv.DictWriter(file, fieldnames=fieldnames)

writer.writeheader()

for produit in produits:

writer.writerow(produit)

*Il faut tout d’abord créer un objet DictWriter  en utilisant la méthode csv.DictWriter  . Cet objet permet d'écrire des dictionnaires dans le fichier CSV en utilisant les noms de champs spécifiés dans fieldnames. Vous pouvez donc au préalable créer une liste avec l’ensemble des noms de l’en-tête, elle constituera la première ligne de votre fichier CSV une fois la méthode writeheader()  appelée. Il ne reste plus qu’à faire une boucle sur l’ensemble de la liste, et écrire chaque dictionnaire de la liste à l’aide de la méthode writerow()  .*

**Correction du code complet :**

from bs4 import BeautifulSoup

import requests

URL = "https://quiz-partie3.oc"

def extract\_data(url):

response = requests.get(url)

html = response.content

return html

def get\_title(soup):

return soup.title.string

def get\_all\_product\_info(soup, products):

for table in soup.find\_all('table'):

for tr in table.find\_all('tr')[1:]:

td = tr.find\_all('td')

name = td[0].get\_text()

description = td[1].get\_text()

price = td[2].get\_text()

quantity = td[3].get\_text()

product = {

'nom': name,

'description': description,

'prix': price,

'quantite': quantity

}

products.append(product)

def convert\_dollar\_to\_euro(products):

change\_rate = 0.8

for product in products:

price\_euro = product['prix'].replace('€', '')

price\_euro = float(price\_euro)

price\_dollar = price\_euro \* change\_rate

product['prix'] = str(round(price\_dollar, 2)) + '$'

def load\_data(soup, products):

title = get\_title(soup)

with open(f'{title}.csv', mode='w', newline='') as file:

fieldnames = ['nom', 'description', 'prix', 'quantite']

writer = csv.DictWriter(file, fieldnames=fieldnames)

writer.writeheader()

for product in products:

writer.writerow(product)

if \_\_name\_\_ == '\_\_main\_\_':

html = extract\_data(url)

soup = Beautiful\_Soup(html, "html.parser")

products = []

get\_all\_product\_info(soup, products)

convert\_dollar\_to\_euro(products)

load\_data(soup, products)