m05_07_beautiful_soup_pratica

April 5, 2021

1 Curso de Python do DS ao DEV

Comunidade DS - Meigarom Lopes

2 Modulo 05 - Extração de Dados em HTML

3 A biblioteca Beautiful Soup - Prática I

[93]:
```
import requests
import pandas as pd

from datetime import datetime

from bs4 import BeautifulSoup
```

[94]:
```
url = 'https://www2.hm.com/en_us/men/products/jeans.html'

headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5)␣
↪AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36'}
page = requests.get( url, headers=headers )
```

[95]:
```
soup = BeautifulSoup( page.text, 'html.parser' )
```

[96]:
```
products = soup.find( 'ul', class_='products-listing small' )
```

[97]:
```
product_list = products.find_all( 'article', class_='hm-product-item')

# product id
product_id = [p.get( 'data-articlecode' ) for p in product_list]

# product category
product_category = [p.get( 'data-category' ) for p in product_list]
```

[98]:
```
# product name
product_list = products.find_all( 'a', class_='link' )
product_name = [p.get_text() for p in product_list]
```

[99]:
```
# price
product_list = products.find_all( 'span', class_='price regular' )
product_price = [p.get_text() for p in product_list]
```

[100]:
```
# product color
```

[101]:
```
# product composition
```

[102]:
```
data = pd.DataFrame( [product_id, product_category, product_name,␣
 ↪product_price] ).T
data.columns = ['product_id', 'product_category', 'product_name',␣
 ↪'product_price']

# scrapy datetime
data['scrapy_datetime'] = datetime.now().strftime( '%Y-%m-%d %H:%M:%S' )
```

[103]:
```
data.head()
```

[103]:
```
   product_id  product_category          product_name product_price  \
0  0636207006    men_jeans_slim            Slim Jeans       $ 19.99
1  0636207010    men_jeans_slim            Slim Jeans       $ 19.99
2  0427159006  men_jeans_skinny  Trashed Skinny Jeans       $ 39.99
3  0720504001  men_jeans_skinny          Skinny Jeans       $ 24.99
4  0690449022  men_jeans_skinny          Skinny Jeans       $ 39.99

       scrapy_datetime
0  2021-04-05 14:08:46
1  2021-04-05 14:08:46
2  2021-04-05 14:08:46
3  2021-04-05 14:08:46
4  2021-04-05 14:08:46
```

4 A biblioteca Beautiful Soup - Prática II

[80]:
```
import numpy as np
```

[68]:
```
url = 'https://www2.hm.com/en_us/men/products/jeans.html'
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5)␣
 ↪AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36'}

page = requests.get( url, headers=headers )
```

[72]:
```
soup = BeautifulSoup( page.text, 'html.parser' )
```

[76]:
```
```

```
total_item = soup.find_all( 'h2', class_='load-more-heading' )[0].get('data-total')
total_item
```

[76]:
```
'94'
```

[83]:
```
page_number = np.round( int( total_item ) / 36 )
page_number
```

[83]:
```
3.0
```

[84]:
```
url02 = url + '?page-size=' + str( int( page_number*36 ) )
url02
```

[84]:
```
'https://www2.hm.com/en_us/men/products/jeans.html?page-size=108'
```

5 A biblioteca Beautiful Soup - Prática III

5.0.1 One Product

[2]:
```
import requests
import pandas as pd

from datetime import datetime

from bs4 import BeautifulSoup
```

[35]:
```
# API Requests
url = 'https://www2.hm.com/en_us/productpage.0636207010.html'

headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36'}
page = requests.get( url, headers=headers )

# Beautiful Soup object
soup = BeautifulSoup( page.text, 'html.parser' )

# ==================== color name ================================
product_list = soup.find_all( 'a', class_='filter-option miniature' )
color_name = [p.get( 'data-color' ) for p in product_list]

# product id
product_id = [p.get( 'data-articlecode' ) for p in product_list]

df_color = pd.DataFrame( [product_id, color_name] ).T
df_color.columns = ['product_id', 'color_name']
```

```
# generate style id + color id
df_color['style_id'] = df_color['product_id'].apply( lambda x: x[:-3] )
df_color['color_id'] = df_color['product_id'].apply( lambda x: x[-3:] )


# ==================== composition ================================
product_composition_list = soup.find_all( 'div',␣
 ↪class_='pdp-description-list-item' )
product_composition = [list( filter( None, p.get_text().split( '\n' ) ) )  for␣
 ↪p in product_composition_list]

# reaname dataframe
df_composition = pd.DataFrame( product_composition ).T
df_composition.columns = df_composition.iloc[0]

# delete first row
df_composition = df_composition.iloc[1:].fillna( method='ffill' )

# generate style id + color id
df_composition['style_id'] = df_composition['Art. No.'].apply( lambda x: x[:-3]␣
 ↪)
df_composition['color_id'] = df_composition['Art. No.'].apply( lambda x: x[-3:]␣
 ↪)

# merge data color + decomposition
data_sku = pd.merge( df_color, df_composition[['style_id', 'Fit',␣
 ↪'Composition']], how='left', on='style_id' )
```

5.0.2 Multiple Products

[135]:
```
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5)␣
 ↪AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36'}

# empty dataframe
df_details = pd.DataFrame()

# unique columns for all products
aux = []

cols = ['Art. No.', 'Composition', 'Fit', 'Product safety', 'Size']
df_pattern = pd.DataFrame( columns=cols )

for i in range( len( data ) ):
    # API Requests
    url = 'https://www2.hm.com/en_us/productpage.' + data.loc[i, 'product_id']␣
 ↪+ '.html'
```

```
    page = requests.get( url, headers=headers )

    # Beautiful Soup object
    soup = BeautifulSoup( page.text, 'html.parser' )

    # ==================== color name ================================
    product_list = soup.find_all( 'a', class_='filter-option miniature' )
    color_name = [p.get( 'data-color' ) for p in product_list]

    # product id
    product_id = [p.get( 'data-articlecode' ) for p in product_list]

    df_color = pd.DataFrame( [product_id, color_name] ).T
    df_color.columns = ['product_id', 'color_name']

    # generate style id + color id
    df_color['style_id'] = df_color['product_id'].apply( lambda x: x[:-3] )
    df_color['color_id'] = df_color['product_id'].apply( lambda x: x[-3:] )

    # ==================== composition ================================
    product_composition_list = soup.find_all( 'div',␣
↪class_='pdp-description-list-item' )
    product_composition = [list( filter( None, p.get_text().split( '\n' ) ) ) ␣
↪for p in product_composition_list]

    # reaname dataframe
    df_composition = pd.DataFrame( product_composition ).T
    df_composition.columns = df_composition.iloc[0]

    # delete first row
    df_composition = df_composition.iloc[1:].fillna( method='ffill' )

    # garantee the same number of columns
    df_composition = pd.concat( [df_pattern, df_composition], axis=0 )

    # generate style id + color id
    df_composition['style_id'] = df_composition['Art. No.'].apply( lambda x: x[:
↪-3] )
    df_composition['color_id'] = df_composition['Art. No.'].apply( lambda x:␣
↪x[-3:] )

    aux = aux + df_composition.columns.tolist()

    # merge data color + decomposition
    data_sku = pd.merge( df_color, df_composition[['style_id', 'Fit',␣
↪'Composition', 'Size', 'Product safety']], how='left', on='style_id' )
```

```
    # all details products
    df_details = pd.concat( [df_details, data_sku], axis=0 )


# Join Showroom data + details
data['style_id'] = data['product_id'].apply( lambda x: x[:-3] )
data['color_id'] = data['product_id'].apply( lambda x: x[-3:] )

data_raw = pd.merge( data, df_details[['style_id', 'color_name', 'Fit',␣
↪'Composition', 'Size', 'Product safety']],
                     how='left', on='style_id' )
```

6 Exercícios Práticos

6.0.1 1. Coletar os seguintes dados da página: https://books.toscrape.com

- Catálogo:
  - Classics
  - Science Fiction
  - Humor
  - Business
- Coletar os seguintes dados de cada livro:
  - Nome do livro
  - Preço em libras
  - Avaliação dos consumidores
  - Disponível em estoque

6.0.2 Entregável:

```
- Faça um plano escrito para cada uma das perguntas de negócio, contendo:
    - Saída: A simulação da tabela e gráfico final.
    - Processo: A sequência de passos organizada pela lógica de execução
    - Entrada: O link para as fontes de dados.

- Uma csv com todas as informação de todos os catálogos.
```

[ ]: