スクレイビング（1）

現代の html 記述には一定のルールがある。そのルールが守られているサイトであれば、その構造を解析することで、html 内の任意の情報を機械的に抽出することができる。

◆Web ページの送受信に使われる HTTP 通信

スクレイビングを実行するためには、Python からサーバに HTTP のリクエストを送り、そのレスポンスから HTML を取得する。

■HTTP 通信

HTTP 通信を行うライブラリが「Requests」[1]。
Requests は Anaconda でインストールされている。

◆書籍ページのレスポンスを取得

```
#1
import requests
res = requests.get('https://book.impress.co.jp/books/1118101169')
res.status_code
```

「200 番台」の数字が返されれば、リクエストが成功している。

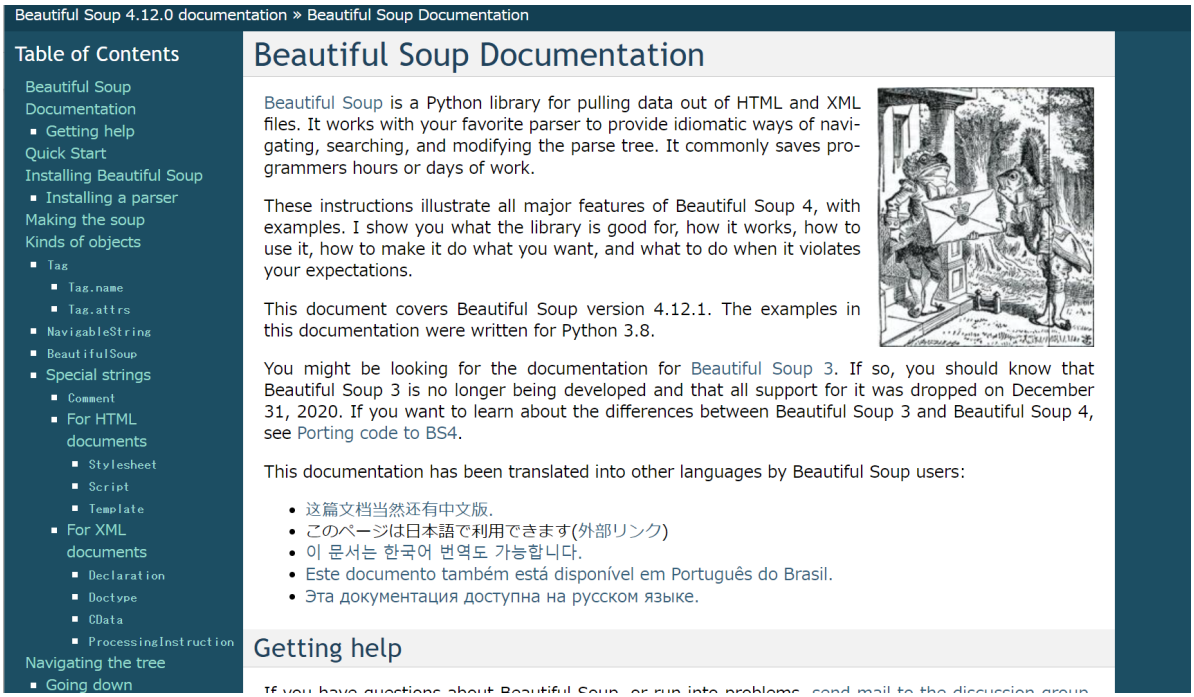
◆レスポンスから HTML を取得

```
#2
html_doc = res.text
print(html_doc[:300])
```

※実行には#1 が必要

[1] Python の標準ライブラリには「urllib.requests」がある

■HTML からデータを抽出 `Beautiful Soup`



Beautiful Soup 4.12.0 documentation » Beautiful Soup Documentation

Table of Contents

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.12.1. The examples in this documentation were written for Python 3.8.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed and that all support for it was dropped on December 31, 2020. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

This documentation has been translated into other languages by Beautiful Soup users:

- 这篇文档当然还有中文版.
- このページは日本語で利用できます(外部リンク)
- 이 문서는 한국어 번역도 가능합니다.
- Este documento também está disponível em Português do Brasil.
- Эта документация доступна на русском языке.

Getting help

If you have questions about Beautiful Soup, or run into problems, send mail to the discussion group

スクレイピング（2）

インプレスブックス社の「すっきりわかる Python 入門」を使ってスクレイピングの実習をします。

https://book.impress.co.jp/books/1118101169

■Beautiful Soup を使う準備

最初に Beautiful Soup のオブジェクトを作成する

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(HTML テキスト、パーサー)
```

第 1 引数：HTML テキスト
第 2 引数：パーサー・・・構文解析を行うプログラム

・Beautiful Soup で仕様可能なパーサー

html_parser　：標準ライブラリに含まれている
lxml　　：非常に高速
html5lib

◆要素を検索　find()/find_all()

`find()`　　：引数の条件で検索して最初に見つけた要素を取得

`find_all()`　：条件に一致する全ての要素を取得。取得した要素はリストで扱える

＜用例＞

`soup.find(class_='not_link')`・・・class 属性で検索

`soup.find('a')`・・・最初の a 要素の検索

`soup.find_all('a')`・・・全ての a 要素の検索

`soup.find(id='abc')`・・・id 属性で要素を検索

`ul_tag = soup.find('ul', id='abc')`・・・id 属性が abc の ul 要素の子孫から

`ul_tag.find('li', class = 'not_link')`・・・class 属性が not_link の li 要素を検索

■ページから要素の取得

書籍のページから「書籍名」と「値段」を取得する

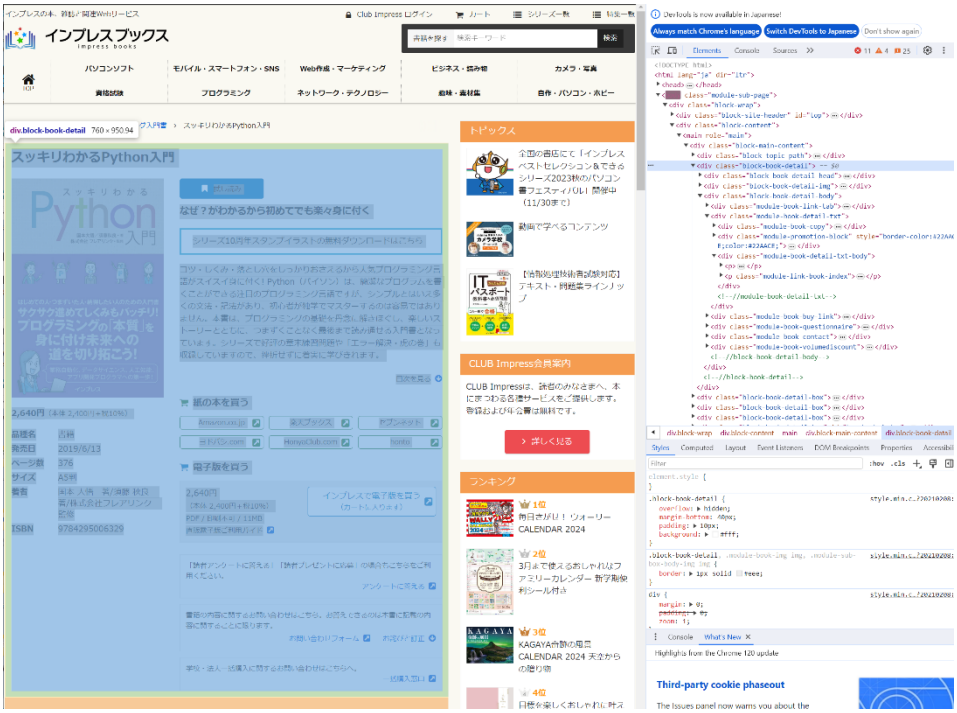
◆タイトルの取得

まず、html の取得

最後の行でオブジェクトを作成している。

```
#1
import requests
from bs4 import BeautifulSoup
res                                                              =
requests.get('https://book.impress.co.jp/books/1118101169')
html_doc = res.text
soup = BeautifulSoup(html_doc, 'html.parser')
```

#2 では書籍情報のブロックを取得。・・・※#1 に続けて入力。



ブラウザ付属のデベロッパーツールで、書籍情報が掲載されているブロックを調べる。書籍情報ブロックは class 属性が `block-book-detail` の div 要素とわかる。

```
#2
div_book_detail = soup.find('div', class_='block-book-detail')
```

#3 では、取得した書籍情報ブロックから書籍名を取得
※＃2 に続けて入力

```
#3
book_title = div_book_detail.find('h2')
book_title.get_text()
```

ここでは、書籍情報ブロックから h2 要素を取得して、そのテキストを取得する
結果（タイトル）が表示される。

◆値段の取得

値段は class 属性が module-book-price の p 要素にある
find で検索し、取得する。

```
#4
book_price = div_book_detail.find('p', class_='module-book-price')
book_price.get_text()
```

・Tag オブジェクト

get_text()メソッドでは、タグ内の文字列が取得できる。

■要素の特定が難しいケース

html が dl、dt、dd などの定義リストで囲まれている場合、取得する情報を特定することができない。

こうした場合、dt 要素と dd 要素でできている組み合わせのデータを Python の辞書データに見立てる。

```
#5
<dl id=’book’>
  <dt>Python 入門</dt><dd>1000 円</dd>
  <dt>スクレイピング本</dt><dd>1800 円</dd>
  <dt>機械学習本</dt><dd>2500 円</dd>
```

Python 辞書データ

```
#6
{
   ‘Python 入門’: ‘1000 円’
   ‘スクレイピング本’: ‘1800 円’
   ‘機械学習本’: ‘2500 円’
}
```

◆要素を絞り込む

html の要素を「辞書データ」に変換して、「発売日」と「著者」を取得する

書籍情報の取得までは（#1、#2）は同じ。

```
#7
dl_book_data  =  div_book_detail.find('dl',  class_='module-
book-data')
book_data = {}
for tag in dl_book_data.find_all(['dt', 'dd']):
    if tag.name == 'dt':
        key = tag.get_text()
    if tag.name == 'dd':
        book_data[key] = tag.get_text().strip()

print('発売日：', book_data['発売日'])
print('著者', book_data['著者'])
```