Onde baixar e instalar:

O Beautiful Soup pode ser baixar a partir do seu gerenciador de pacotes, por exemplo pode se usar o pip que e um gerenciador nativo do python ou ate mesmo usar o gerenciador do ubuntu que tem as mesma funcionalidades, podendo funcionar no python 2 ou 3, caso essas opcoes nao sejam viaveis sera possivel baixar o mesmo em https://www.crummy.com/software/BeautifulSoup/bs4/download/4.0/.
Os comandos a seguir representam a maneira informada na documentação oficial da ferramenta, vejamos o passo a passo a seguir: pip install BeautifulSoup, a maneira mais simple de se instalar a ferramenta, para usar basta seguir o exemplo a seguir:

Para esse exemplo vamos coletar e tratar um dado a partir de uma página html

Exemplo de como usar:

```
1  from bs4 import BeautifulSoup
2  import requests
3
4  req = requests.get('https://mangahosted.com/')
5  soup = BeautifulSoup(req.content, 'html.parser')
6  print('foi')
7  print(soup.prettify())
8
```

A partir deste código conseguimos obter o conteúdo da página de destino onde req é a requisição e soup e a nossa ferramenta fazer a coleta de dados, o ultimo print significa que o código vai ser formatado de um jeito que quem está programando vai entender e interpretar o mesmo e usar para fazer os testes necessários.

logo abaixo temos um exemplo do prettify, ferramenta própria do beautifulsoup:

```
 <meta charset="utf-8"/>
 <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
 <meta content="IE=Edge,chrome=1" http-equiv="X-UA-Compatible"/>
 <meta content="noindex, nofollow" name="robots"/>
 <meta content="width=device-width,initial-scale=1" name="viewport"/>
 <link href="/cdn-cgi/styles/main.css" id="cf_styles-css" media="screen,projection" rel="stylesheet" type="text/css"/>
 <script defer="" src="https://api.radar.cloudflare.com/beacon.js">
 </script>
</head>
<body>
 <div id="cf-wrapper">
  <div class="cf-alert cf-alert-error cf-cookie-error hidden" data-translate="enable_cookies" id="cookie-alert">
   Please enable cookies.
  </div>
  <div class="p-0" id="cf-error-details">
   <header class="mx-auto pt-10 lg:pt-6 lg:px-8 w-240 lg:w-full mb-15 antialiased">
    <h1 class="inline-block md:block mr-2 md:mb-2 font-light text-60 md:text-3xl text-black-dark leading-tight">
     <span data-translate="error">
      Error
     </span>
     <span>
      1020
     </span>
    </h1>
    <span class="inline-block md:block heading-ray-id font-mono text-15 lg:text-sm lg:leading-relaxed">
     Ray ID: 6b86bf699f72798c •
    </span>
    <span class="inline-block md:block heading-ray-id font-mono text-15 lg:text-sm lg:leading-relaxed">
     2021-12-04 17:27:50 UTC
    </span>
    <h2 class="text-gray-600 leading-1.3 text-3xl lg:text-2xl font-light">
     Access denied
    </h2>
   </header>
   <section class="w-240 lg:w-full mx-auto mb-8 lg:px-8">
    <div class="w-1/2 md:w-full" id="what-happened-section">
     <h2 class="text-3xl leading-tight font-normal mb-4 text-black-dark antialiased" data-translate="what_happened">
      What happened?
     </h2>
     <p>
      This website is using a security service to protect itself from online attacks.
     </p>
    </div>
   </section>
   <div class="cf-error-footer cf-wrapper w-240 lg:w-full py-10 sm:py-4 sm:px-8 mx-auto text-center sm:text-left border-solid border-0 border-t border-gray-300">
    <p class="text-13">
     <span class="cf-footer-item sm:block sm:mb-1">
      Cloudflare Ray ID:
      <strong class="font-semibold">
       6b86bf699f72798c
      </strong>
     </span>
     <span class="cf-footer-separator sm:hidden">
```

Com a imagem assim podemos ter uma noção de como se usar a ferramenta na prática para os próximos tópicos iremos abordar algo mais concreto onde iremos usar funções para realizar os testes.

características

O BeautifulSoup é uma biblioteca python de coleta de dados de arquivo html e xml, onde interpreta o melhor parser (conversão direta para algo) e economiza algumas horas do nosso dia de trabalho.
Podemos trazer dados para a tela da página de destino, buscar apenas elementos específicos, onde é otimizado o processo, pois como a ferramenta usar apenas busca por texto e traz resultados desejados.

teste dinâmico com sintaxe e caso de teste

Para fechar vamos fazer um caso de teste abordando sintaxe e performance onde iremos demonstrar uma comparação entre usar o código em selenium e usar a mesma captura de dados com beautifulsoup .

Lo abaixo podemos visualizar o código que foi desenvolvido em python:

```python
from bs4 import BeautifulSoup
import requests
import time
import time
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
from bs4 import BeautifulSoup, element
import pandas as pd
import numpy as np
import logging



def findElementBySoup():
    req = requests.get('https://mangahosted.com/')
    soup = BeautifulSoup(req.content, 'html.parser')
    soup.find_all('a')

def findElementBySelenium():

    #options = webdriver.ChromeOptions()
    options = Options()
    options.add_argument("--headless")
    options.add_argument("--window-size=1920,1200")
    options.add_argument("--disable-dev-shm-usage")
    options.add_argument("--disable-popup-blocking")
    options.add_argument("--disable-web-security")

    options.add_argument("--no-sandbox")
    chrome_prefs = {}
    options.experimental_options["prefs"] = chrome_prefs
    chrome_prefs["profile.default_content_settings"] = {"images": 2}


    driver = webdriver.Chrome("C:/Users/biela/Downloads/chromedriver.exe", options=options)
    driver = webdriver.Chrome(options=options)

    driver.get("https://mangahosted.com/")

    html_msc = driver.page_source
    soup_msc = BeautifulSoup(html_msc, 'html.parser')
    table_msc = soup_msc.find_all('a')
    return "feito"

ini = time.time()
findElementBySelenium()
fim = time.time()
print ("findElementBySelenium ", fim-ini)

ini1 = time.time()
findElementBySoup()
fim1 = time.time()
print ("findElementBySoup ", fim1-ini1)
```

De cara percebemos a diferença entre atribuições de variáveis e de sintaxe, onde um código e muito mais na versão selenium em relação ao código em beautifulsoup.
Selenium por si só é uma ferramenta muito robusta mas que exige muito de uma máquina comum, sendo assim o beautifulsoup entra para melhorar a busca de dados.
Por fim obtermos o tempo de execução de cada um:
Selenium 10.023021459579468
Soup 0.11469221115112305

Apresentado os resultados percebemos a diferença de valores em relação um ao outro, em casos de repetições exaustivas selenium pode ser ainda mais pesado por existe um acúmulo de atividades que ficam em aguardo até uma outra atividade terminar, sendo assim tornando a ferramenta útil apenas para processos menos repetitivos.