Tópicos Especiais em Sistemas de Informação

Raspagem de dados com Requests e Beautiful Soap

Ely – elydasilvamiranda@gmail.com

Fonte

- Slides baseados no curso:
  - Getting Started with Python Web Scraping - Charles Clayton
  - https://www.packtpub.com/mapt/video/big_data_and_business_intelligence/9781787283244
  - Livro Python Web Scrapping: fetching data from the WEB, 2a. Ed. de Katharine Jarmul, Richard Lawson.
  - Alguns slides usam como exemplo o site:
    - http://example.webscraping.com/

elydasilvamiranda@gmail.com

Raspagem de dados

- Do inglês "web scraping";
- Processo de coleta de dados;
- Aplicado a dados:
  - Não facilmente disponíveis;
  - Não disponíveis por uma API;
  - Em formatos não tabulados, como PDF;

elydasilvamiranda@gmail.com

Algumas situações

- Comparações de preço;
- Pesquisar notícias;
- Alertas e monitoramento de variações de preço;
- Monitoramento de mercadorias;
- Acompanhamento de índices financeiros;
- Automatização de tarefas tediosas, como publicações em diário oficial;
- Atualizações de páginas em geral.

elydasilvamiranda@gmail.com

Meios para raspagem de dados

- Pelas ferramentas do próprio navegador:
  - Para pequenas quantidades de dados em uma página;
- Por web drivers:
  - Uma alternativa para quando se possuem várias páginas ou seja necessário simular cliques em componentes
- Por APIs de bibliotecas HTTP:
  - Quando for necessário serviços em background e um controle a mais de toda a página;
- Iniciaremos nossos estudos por web drivers e nas próximas aulas focaremos em APIs do Python.

aydiasantana@gmail.com

Bibliotecas necessárias

- Requests:

  > pip install requests

- Beautiful Soup:

  > pip install beautifulsoup4

- Html5lib

  > pip install html5lib

elydasilvamiranda@gmail.com

Requisições HTTP

- Requisições via web driver não são performáticas:
  - Uma instância do navegador ficar aberta, ainda que em headless mode;
  - Há um gap de comunicação entre os scripts e o web driver;
- Para raspar dados de uma página, deve-se preferencialmente baixá-la via HTTP;
- A forma mais simples de baixar uma página é utilizar bibliotecas que façam requisições HTTP;

elydasilvamiranda@gmail.com

Requests

- Para fazer requisições HTTP com Python, há várias bibliotecas;
- As que mais se destacam são:
  - httplib (http.client) e urllib: nativas do Python, porém muito verbosas e com algumas limitações;
  - Requests: implementação independente, mas não verbosa com várias características relevantes;

elydasilvamiranda@gmail.com

Requests

- "HTTP para Humanos";
- Documentação:
  - http://docs.python-requests.org/pt_BR/latest/
- Instalação:
  - pip install requests

elydasilvamiranda@gmail.com

Exemplo 1

- Baixando uma página:

```
# -*- coding: utf-8 -*-
import requests
response = requests.get('http://www.google.com')
print(response.status_code)
print(response.headers['content-type'])
print(response.text)
```


elydasilvamiranda@gmail.com

Exemplo 2

- Tratando exceções:

```
import requests

def download(url, num_retries=2):
    print('Downloading:', url)
    page = None
    try:
        response = requests.get(url)
        page = response.text
    except requests.exceptions.RequestException as e:
        print('Download error:', e.reason)
    return page

# testing...
page = download('http://www.google.com/')
print(page)
```


elydasilvamiranda@gmail.com

Exemplo 3

- Tratando erros HTTP e fazendo mais de uma tentativa:
  - Caso o código de status seja um dos "400", exibimos o erro;
  - Caso o erro esteja na faixa dos "500", tentamos novamente.

elydasilvamiranda@gmail.com

Exemplo 3

```
import requests

def download(url, num_retries=2):
    print('Downloading:', url)
    page = None
    try:
        response = requests.get(url)
        page = response.text
        if response.status_code >= 400:
            print('Download error:', response.text)
            if num_retries and 500 <= response.status_code < 600:
                return download(url, num_retries - 1)
    except requests.exceptions.RequestException as e:
        print('Download error:', e.reason)
    return page

# testing...
page = download('http://www.google.com/')
print(page)
```


elydasilvamiranda@gmail.com

Parsing em HTML

- Regular expressions:
  - Já faz parte da distribuição Python;
  - https://docs.python.org/3/howto/regex.html
- Beautiful Soap:
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Lxml:
  - Requer uma instalação mais complexa;
  - Disponível na distribuição Anaconda;
  - http://lxml.de/installation.html

elydasilvamiranda@gmail.com

Parsing em HTML

Abordagem	Performance	Facilidade uso	Instalação
Regular expressions	Rápida	Difícil	Fácil (built-in)
Beautiful Soup	Lenta	Fácil	Fácil (pip)
Lxml	Rápida	Fácil	moderada

elydasilvamiranda@gmail.com

Estudo de caso

- Dado país:
  - http://example.webscraping.com/places/default/view/United-Kingdom-239
- Extrair a área:

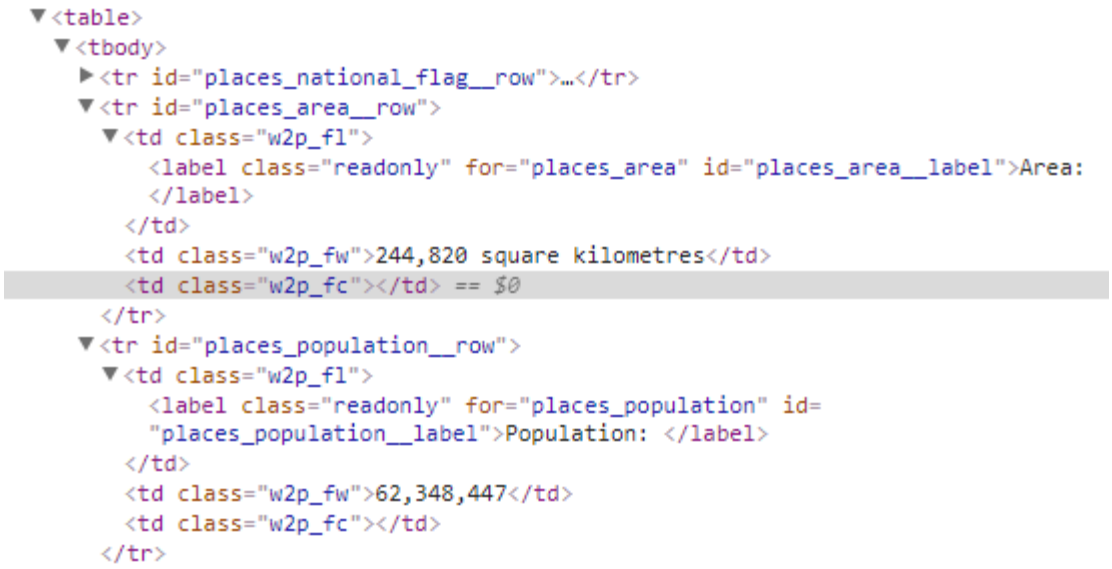
Example web scraping website

National Flag:	
Area:	244,820 square kilometres
Population:	62,348,447

elydasilvamiranda@gmail.com

Estudo de caso

- Inspecionando o código fonte:
  - Há uma tabela com os dados do país;
  - Há várias colunas, mas o dado está em td's com `w2p_fw;`
  - Infelizmente, há várias dessas colunas.



elydasilvamiranda@gmail.com

Estudo de caso

- Com regular expressions:

```
import re

url = 'http://example.webscraping.com/places/default/view/United-Kingdom-239'
page = download(url)
area = re.findall(r'<td class="w2p_fw">(.*?)</td>',
          page)[1]
# alternativa:
# area re.findall('''<tr
id="places_area__row">.*?<tds*class=[''']w2p_fw[''']>
   (.*?)</td>''', html)

print(area)
```


elydasilvamiranda@gmail.com

Estudo de caso

- Com Beautiful Soap:

```
from bs4 import BeautifulSoup
url =
'http://example.webscraping.com/places/default/view
/United-Kingdom-239'
html = download(url)
soup = BeautifulSoup(html, 'html.parser')
tr = soup.find(attrs={'id':'places_area__row'})
td = tr.find(attrs={'class':'w2p_fw'})
area = td.text
print(area)
```


elydasilvamiranda@gmail.com

Parsers para Beautiful Soup

- html.parser:
  - Vem com a distribuição padrão do Python;
  - Possui algumas limitações para interpretar HTML mal formado;
- html5lib:
  - Instalação via pip;
  - Cria um html 5 válido corrigindo erros de má formação.

elydasilvamiranda@gmail.com

Parsers para Beautiful Soup

- Muitas páginas não possuem HTML bem formado, como no exemplo abaixo:

```
<ul class=country>
 <li>Area
     <li>Population
</ul>
```

- Para um parser, há a dúvida se população é um filho de área ou um outro item da lista;
- O parser 'html.parser' padrão não lida bem com essa situação;
- Uma alternativa é usar o parser **html5lib**

elydasilvamiranda@gmail.com

Parsers para Beautiful Soup

- Usando o html.parser:

```
from bs4 import BeautifulSoup
from pprint import pprint
broken_html =
 '<ul class=country><li>Area<li>Population</ul>'
soup = BeautifulSoup(broken_html, 'html.parser')
fixed_html = soup.prettify()
pprint(fixed_html)
```

```
('<ul class="country">\n'
 '    <li>\n'
 '        Area\n'
 '        <li>\n'
 '            Population\n'
 '        </li>\n'
 '    </li>\n'
 '</ul>')
```


elydasilvamiranda@gmail.com

Parsers para Beautiful Soup

- Usando o parser html5lib:

```
...
soup = BeautifulSoup(broken_html, 'html5lib')
...
```

```
'<html>\n'
' <head>\n'
' </head>\n'
' <body>\n'
'  <ul class="country">\n'
'   <li>\n'
'    Area\n'
'   </li>\n'
'   <li>\n'
'    Population\n'
'   </li>\n'
'  </ul>\n'
' </body>\n'
'</html>')
```


elydasilvamiranda@gmail.com

Métodos find

- O BS usa os métodos find e find_all para obter trechos do HTML
  - find: trás a primeira ocorrência encontrada;
  - find_all: trás todas as ocorrências;

https://www.crummy.com/software/BeautifulSoup/bs4/doc/#find

https://www.crummy.com/software/BeautifulSoup/bs4/doc/#find-all

elydasilvamiranda@gmail.com

Exemplos

- soup.find_all('p')
  - retorna todos os parágrafos;
- soup.find_all(['th','td'])
  - retorna todas as tags de uma lista, no caso, th e as tags td;
- soup.find_all(class_ = 'buzz')
  - retorna todos os elementos que possuem o atributo class igual a 'buzz';
- soup.find_all(href = 'http://www.ifpi.edu.br')
  - retorna os elementos que possuem o atributo href http://www.ifpi.edu.br;
- soup.find_all(id = re.compile(r'^foo'))
  - retorna os elementos que possuem o id começando com 'foo'.

elydasilvamiranda@gmail.com

Métodos select

- É possível dados de uma página HTML usando seletores CSS:
  - select_one: trás a primeira ocorrência encontrada;
  - select: trás todas as ocorrências.

https://www.crummy.com/software/BeautifulSoup/bs4/doc/#css-selectors

elydasilvamiranda@gmail.com

Exemplos

- soup.select('p')
  - retorna todos os parágrafos;
- soup.select('th, td')
  - retorna todas as tags de uma lista, no caso, th e as tags td;
- soup.select('.buzz')
  - retorna todos os elementos que possuem o atributo class igual a 'buzz';
- soup.select('[href = http://www.ifpi.edu.br]')
  - retorna os elementos que possuem o atributo href http://www.ifpi.edu.br;
- soup.find_all('[id^=foo]')
  - retorna os elementos que possuem o id começando com 'foo'.

elydasilvamiranda@gmail.com

Questões legais

- E se quiséssemos as 1000 primeiras páginas?
- Os efeitos de raspagens de forma irresponsáveis podem ser danosos;
- Raspagens de dados agressivas podem comprometer serviços.

elydasilvamiranda@gmail.com

Questões legais

- Teoricamente não há problemas se:
  - Se os dados estão disponíveis e ...
  - Seu uso for apenas pessoal e ...
  - Não há violação de direitos e ...
  - Há preocupação com a frequência e forma das requisições.

elydasilvamiranda@gmail.com

Questões legais

- Porém, devem-se observar questões éticas e legais se:
  - A raspagem envolve sobrecarregar sites;
  - Os dados serão republicados e redistribuídos e tiverem direitos autorais envolvidos;
  - Forem feitos cruzamentos de dados isolados, expondo informações novas a partir disso;

- Além disso, sites são negócios e "raspadores" não provêem receita aos sites.

elydasilvamiranda@gmail.com

Casos conflituosos

- Alguns casos em que raspagem de dados geraram conflitos:
  - http://caselaw.lp.findlaw.com/scripts/getcase.pl?court=US&vol=499&invol=340
  - http://www.austlii.edu.au/au/cases/cth/FCA/2010/44.html
  - http://www.nysd.uscourts.gov/cases/show.php?db=special&id=279
  - http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf
  - https://www.paed.uscourts.gov/documents/opinions/16D0129P.pdf

elydasilvamiranda@gmail.com

Consequências

- De toda forma, ao raspar dados de um site, o registro do acesso pode ser monitorado;
- Em casos de "má conduta", poderá haver:
  - bloqueio de endereços;
  - Questões judiciais relacionadas à direitos autorais;
  - Surgimento de captchas.

elydasilvamiranda@gmail.com

Sugestões

- Faça raspagens responsáveis e éticas:
- Introduza delay nos scripts, criando pausas entre as requisições;
- Rode scripts em horários que não haja picos de acesso;
- Leia, se disponível, temos e condições para raspagem, mineração, extração e etc;
- Peça permissão ao administrador do site, talvez o próprio disponibilize os dados de outra forma;
- Se possível, use uma API.

elydasilvamiranda@gmail.com

Tópicos Especiais em Sistemas de Informação

Raspagem de dados com Requests e Beautiful Soap

Ely – elydasilvamiranda@gmail.com