Scraping

Ali Akbar Septiandri

Universitas Al-Azhar Indonesia

aliakbars@live.com

October 3, 2019

Overview

Bahan Bacaan

1. Dokumentasi Requests: `http://docs.python-requests.org/en/master/`
2. Dokumentasi Beautiful Soup: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`
3. Dokumentasi Scrapy: `https://doc.scrapy.org/en/1.3/`

Scraping

Pendahuluan

- Informatika tidak jauh dengan data dan informasi
- Proses pengumpulan data tidak selalu mudah
- Apakah datanya boleh dibagikan?

Pengelolaan Data

- Untuk membuat sebuah perangkat lunak, terkadang kita perlu menggunakan protokol atau kakas yang telah disediakan oleh orang lain
- Kita perlu data yang sudah disediakan orang lain untuk dianalisis, e.g. analisis sentimen dari *tweets* di Twitter
- Ada situs yang langsung menyediakan dataset, ada situs yang menyediakan API

Application Programming Interface (API)

Beberapa situs yang menyediakan API:

1. Facebook (`https://developers.facebook.com/`)
2. Twitter (`https://dev.twitter.com/overview/api`)
3. Instagram (`https://www.instagram.com/developer/`)
4. DuckDuckGo (`https://api.duckduckgo.com/api`)

Sumber Data

Beberapa situs yang menyediakan data yang sudah siap diolah:

1. Kaggle (`https://www.kaggle.com/datasets`)
2. UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/datasets.html`)
3. Portal Data Indonesia (`http://data.go.id/`)
4. SNAP (`http://snap.stanford.edu/`)

Sumber Data

Beberapa situs tidak menyediakan dataset atau API untuk memberikan data karena:

1. tidak dikembangkan sejak awal;
2. tidak ingin datanya disebarkan, e.g. Instagram; atau
3. hanya bisa diakses terbatas, e.g. Microdata BPS

sehingga **mungkin** perlu dilakukan *scraping*.

“visible $\neq$ accessible $\neq$ storable $\neq$ presentable”
(Lavrenko, 2010)

Requests

Requests is an elegant and simple HTTP library for Python, built for human beings.

Sekilas tentang Requests

Example

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

HTTP Methods

Requests mendukung semua HTTP methods yang ada:

1. GET
2. POST
3. PUT
4. DELETE
5. HEAD
6. OPTIONS

Response Content

- Secara bawaan, HTTP requests akan menghasilkan kembalian berupa teks
- Teks *response content* pun bisa dalam berbagai format, e.g. HTML, XML, JSON
- Ada beberapa kasus saat *response content*-nya berupa *binary*, e.g. gambar, suara

Semua sudah ditangani Requests!

HTTP POST

Example (Mengisi form)

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}

>>> r = requests.post("http://httpbin.org/post", data=payload)
>>> print(r.text)
```

Example (Mengunggah file)

```
>>> url = 'http://httpbin.org/post'
>>> files = {'file': open('report.xls', 'rb')}

>>> r = requests.post(url, files=files)
>>> r.text
```

Response Status Codes

Example (Kasus berhasil)

```
>>> r = requests.get('http://httpbin.org/get')
>>> r.status_code
200
```

Example (Kasus error)

```
>>> bad_r = requests.get('http://httpbin.org/status/404')
>>> bad_r.status_code
404

>>> bad_r.raise_for_status()
Traceback (most recent call last):
  File "requests/models.py", line 832, in raise_for_status
    raise http_error
requests.exceptions.HTTPError: 404 Client Error
```

Basic Auth

Tidak semua API bisa dibuka begitu saja, beberapa memerlukan otentikasi. Salah satu metode yang umum adalah **Basic Auth**.

Example

```
>>> from requests.auth import HTTPBasicAuth
>>> auth = HTTPBasicAuth('fake@example.com', 'not_a_real_password')

>>> r = requests.get(url=url, auth=auth)
>>> r.status_code
201
```

Penguraian HTML

Scraping

- Pada prinsipnya, setiap hal yang terlihat di peramban web (*web browser*) bisa di-*scrape*
- Yang perlu dilakukan hanya mengambil berkas HTML, lalu mengurainya
- Pustaka Python untuk ini: **Beautiful Soup**

Beautiful Soup

You didn't write that awful page.
You're just trying to get some data out of it.
Beautiful Soup is here to help.
Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Contoh Dokumen HTML

Example (alice.html)

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters;
and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

Quick Start

Example

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(open('alice.html'), 'html.parser')
>>>
>>> print(soup.prettify())
```

Objek Hasil Penguraian

Terdapat empat objek sebagai hasil dari penguraian (*parsing*) dengan Beautiful Soup:

- Tag
- NavigableString
- BeautifulSoup
- Comment

Perhatikan bahwa hasil penguraian ini dapat dilihat sebagai struktur data pohon!

Navigasi

Ke bawah

1. `.contents` (list) dan `.children` (iterator)
2. `.descendants`
3. `.string`, `.strings`, dan `.stripped_strings`

Ke atas

1. `.parent`
2. `.parents`

Navigasi

Ke samping

1. `.next_sibling` dan `.previous_sibling`
2. `.next_siblings` dan `.previous_siblings` (generator)

Ke depan-belakang

1. `.next_element` dan `.previous_element`
2. `.next_elements` dan `.previous_elements` (generator)

Filters

Ada beberapa filters yang bisa digunakan ke metode seperti `find_all()`, antara lain:

1. string
2. regular expression
3. list
4. fungsi (*higher-order functions*)
5. `True`

Pencarian

1. `find_all()`
2. `find()`
3. `find_parents()` dan `find_parent()`
4. `find_next_siblings()` dan `find_next_sibling()`
5. `find_previous_siblings()` dan `find_previous_sibling()`
6. `find_all_next()` dan `find_next()`
7. `find_all_previous()` dan `find_previous()`

CSS Selectors

Beautiful Soup mendukung pencarian dengan CSS selectors melalui metode `select()` dan `select_one()`.

Example

```
soup.select("title")
# [<title>The Dormouse's story</title>]

soup.select("body a")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie"  id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select("head > title")
# [<title>The Dormouse's story</title>]

soup.select(".sister")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

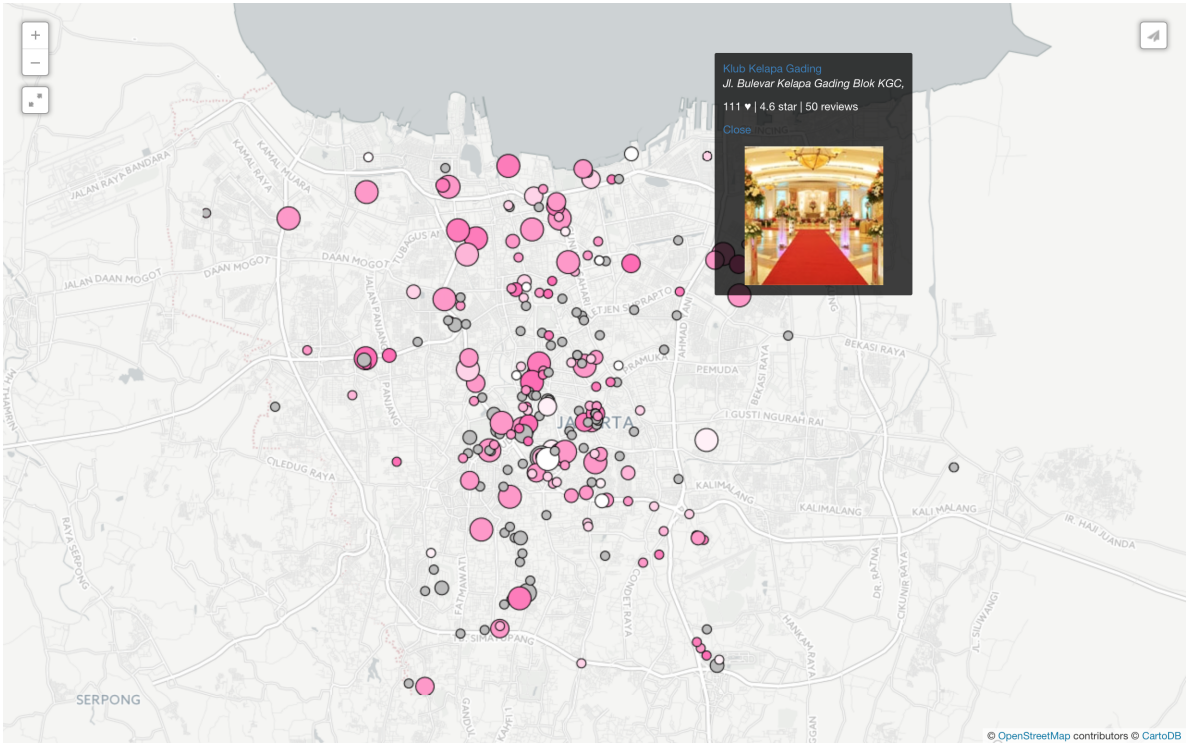
CSS Selectors

Example

```
soup.select("#link1")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]

soup.select('a[href="http://example.com/elsie"]')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]
```

Contoh Aplikasi



Gambar: Pemetaan tempat resepsi di Jakarta dari hasil *scraping* situs `http://www.weddingku.com/` (Wibisono, 2015)

*Kode tersedia di GitHub

Scrapy

Scrapy

An open source and collaborative framework
for extracting the data you need from websites.
In a fast, simple, yet extensible way.

Web Spiders

Example (myspider.py)

```
import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            'http://quotes.toscrape.com/page/1/',
            'http://quotes.toscrape.com/page/2/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

Beberapa Hal yang Sudah Disediakan

- Pencatatan statistik
- Modul pengiriman e-mail
- Simulasi login

What If?

Gambar: `http://what-if.xkcd.com/`

Serious scientific answers to absurd hypothetical questions

Terima kasih