

Screen-scraping with Python

An introduction

Aaron Bycoffe

757 Python Users Group • April 2010



Why screen-scrape?

- Most data on the web isn't available in an easy-to-download format.
- Government bureaucracies can sometimes be less than pleasant to work with.
- For information that changes often, it's easier than requesting it over and over.

Skills screen-scraping can help build

- HTTP request/response cycle.
- Regular expressions.
- Ingenuity.

Some caveats

- Don't steal copyright content.
- Check the site's terms of service.
- Be honest.
- Use common sense.

An example

- **Norfolk outstanding warrants.**
(http://norfolk.gov/Police/wanted_person.asp)

An example

The screenshot shows a web browser window displaying the Norfolk Police Department's 'Outstanding Warrants' page. The browser's address bar shows the URL http://norfolk.gov/Police/wanted_person.asp. The page header includes the Norfolk Police Department logo and navigation links for Home, Residents, Visitors, Business, City Hall, Education, News & TV, Jobs, A to Z, and Staff. A search bar is located in the top right corner.

The main content area features a date stamp: **Tuesday, 04-20-2010**. Below this, a paragraph explains that users can find names of individuals with outstanding warrants by selecting a letter corresponding to their last name. It also provides contact information for CRIME LINE at 1-888-LOCK-U-Up (1-888-562-5887) and advises that if a name is not found, the person either never had a warrant or it has been served.

A list of instructions follows, advising users not to attempt arrests and to provide specific information when calling:

1. The name of the person who is wanted.
2. The charges specified in the warrant.
3. The location the person who is wanted.
4. Any clothing description of the person who is wanted.
5. If the person is driving a vehicle, the make, model, color of the vehicle, and most importantly, the license plate.

Below the instructions is a grid of letters for selection:

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	Q	R	S	T
U	V	W	X	Y
Z				

At the bottom of the page, there is a banner for the Crime Line Program with the text: "The Norfolk Police Department is proud to be partnering with the Crime Line Program in this project." Below the banner is the slogan "CRIME LINE IT WORKS. IT PAYS." and the phone number "1-888-562-5887". A final line of text reads: "This is where Norfolk Crime Line Work Best!"

An example

Chrome File Edit View History Bookmarks Window Help 59° (71%)

Outstanding Warrants - Warn X

http://norfolk.gov/Police/wanted_person_details.asp?letter=A

Home

Wanted Persons - A
(Last names highlighted in blue have an associated picture for the wanted person.)

NAME	CHARGE	SEX	RACE	DOB	HGT	WGT	EYE	HAIR	ISSUE DATE
ABDULLAH MUHAMMAD	FAILURE TO APPEAR FOR TRAFFIC SUMMONS	M	B	7/17/1950	506	200	BRO	BLK	4/10/2009
ABEL KEVIN DUJUAN	BREAKING & ENTER W/INT TO COMMIT FELONY	M	B	7/21/1985	511	174	BRO	BRO	10/10/2006
ABELEDA BENJAMIN	CODE VIOLATIONS PROHIBITED	M	U						2/17/2010
ABELEOA BENJAMIN	CODE VIOLATIONS PROHIBITED	M	U						4/16/2010
ABERDEEN WINSTON M	FAILURE TO APPEAR FOR TRAFFIC SUMMONS	M	B	4/18/1977	505	140	BRO	BLK	1/24/2008
ACOSTA ARTHUR T	FAIL TO APPEAR ON MISDEMEANOR CHARGE	M	W	11/15/1985					1/17/2007
ADACH CHRISTOPHER	GRAND LARCENY	M	W	1/6/1981	600	175		BRO	1/29/2009
ADAM PAULINE BYRD	ATTEMPTED FELONY VIOLATIONS OF DRUG ACT	F	B	12/12/1967	501	105	BRO	BLK	1/31/2007
Adams Barbara L	FRAUDULENT CONV, REMOVAL LEASED PROPERTY >=\$200	F	B	6/21/1964	58	200			10/7/2009
ADAMS CHESTER JERMAINE JR	CONTEMPT OF COURT	M	B	3/27/1984	509	240	BRO	BLK	8/13/2009
ADAMS DARREN TOBIAS	FAIL TO APPEAR ON MISDEMEANOR CHARGE	M	B	9/25/1977	601	242	BRO	BLK	11/20/2009
ADAMS DORIS	ISSUING BAD CHECKS <\$200	U	U						5/31/2007
ADAMS JAMES THOMAS III	FAILURE TO APPEAR FOR TRAFFIC SUMMONS	M	B	3/21/1984	507	175	BLK	BLK	1/17/2007
ADAMS SHAMONIQUE RONNELLE	FAIL TO APPEAR ON MISDEMEANOR CHARGE	F	B	2/20/1982	504	135	BRO	BLK	4/24/2009
ADDAI THYRA SEFAH	REVOCAION OF SUSPENDED SENTENCE AND PROBATION	F	B	9/15/1988	508	120	BRO	BLK	9/8/2008
ADDISON TONY OWENS	FAIL TO COMPLY WITH SUPPORT OBLIGATIONS	M	B	7/20/1972	506	175	BRO	BLK	3/30/2009
ADKINS JOHN LAVONE	REVOCAION OF SUSPENDED SENTENCE AND PROBATION	M	B	8/21/1986					1/13/2009
ADLER MONTE ALLEN	CONTEMPT ORDER CAPIAS	M	W	9/6/1966	602	195	BLU	BRO	1/19/1993
ADRIAN ADONA ENIKA	FAIL TO APPEAR ON MISDEMEANOR CHARGE	F	B	11/26/1984	504	128	BRO	BLK	2/21/2007
AGUILAR ELMER	FAILURE TO APPEAR FOR TRAFFIC SUMMONS	M	W	11/23/1987					10/20/2009
AIKENS NATHAN	TRESPASSING	M	B						11/10/2008

will be saved as a PNG file on your desktop. (The file is saved as PDF in Mac OS 10.3 and

Issues with the site

- No search.
- No sorting.
- No filtering (except by first letter of last name).
- No way to know when data is updated.
- Can't download raw data for analysis.

Let's begin

```
>>> from urllib2 import urlopen  
>>> import re
```

urllib2.urlopen

```
>>> from urllib2 import urlopen
```

- `urllib2.urlopen(url[, data][, timeout])`

“Open the URL url, which can be either a string or a Request object.”

<http://docs.python.org/library/urllib2.html>

re

```
>>> import re
```

- “This module provides regular expression matching operations similar to those found in Perl.”

<http://docs.python.org/library/urllib.html>

Request/response

```
>>> url = 'http://norfolk.gov/Police/  
wanted_person.asp'
```

```
>>> response = urlopen(url)
```

```
>>> response.code  
200
```

```
>>> response.headers.dict['x-powered-by']  
'ASP.NET'
```

Request/response

```
>>> content = response.read()
```

```
>>> print content
```

```
<html>
```

```
<head>
```

```
<title>Outstanding Warrants - Wanted Person:
```

```
Department of Police - The City of Norfolk, VA</
```

```
title>
```

```
<meta http-equiv="Content-Type" content="text/  
html; charset=iso-8859-1">
```

```
...
```

Getting what we want

```
>>> # Find all the strings on this page that match  
... # the pattern of a letter detail page URL.
```

```
>>> urls = re.findall(r'wanted_person_details.asp  
\?letter=[A-Z]', content)
```

```
>>> urls  
['wanted_person_details.asp?letter=A',  
'wanted_person_details.asp?letter=B',  
'wanted_person_details.asp?letter=C', ...]
```

Getting what we want

```
>>> # These are relative URLs. We need to add  
# the base URL to them.
```

```
>>> urls = ['http://norfolk.gov/Police/%s' % url for  
url in urls]
```

```
>>> urls[0]  
'http://norfolk.gov/Police/  
wanted_person_details.asp?letter=A'
```

Getting what we want

```
>>> # Loop through the URLs and get the  
# content of each letter page.
```

```
>>> for url in urls:  
    page = urlopen(url).read()  
    # Parse the page to get the data
```

Parsing the page

```
>>> table = re.search(r'<table width="780"
border="1" cellpadding="2" cellspacing="0"
style="border-color:#CCCCCC; border-width:
1px;" >(.*?)</table>', page, re.S)
```

```
rows = re.findall('<tr>(.*?)</tr>', table.group(),
re.S)
```

Parsing the page

```
>>> fields = ['last', 'first', 'charge', 'sex', 'race',  
'dob', 'height', 'weight', 'eye', 'hair', 'issue_date',]
```

```
>>> data = []
```

```
>>> for row in rows[1:]:  
    rowdata = re.findall(r'<td>(.*?)</td>',  
row)  
    rowdata = dict(zip(fields, rowdata))  
    data.append(rowdata)
```

Parsing the page

```
>>> data[0]
{'issue_date': '4/10/2009', 'last': 'ABDULLAH',
 'weight': '200', 'dob': '7/17/1950', 'sex': 'M', 'hair':
 'BLK', 'charge': 'FAILURE TO APPEAR FOR
 TRAFFIC SUMMONS', 'race': 'B', 'eye': 'BRO',
 'height': '506', 'first': 'MUHAMMAD'}
```

Putting it all together

```
def parse_letter_page(page):
    fields = ['last', 'first', 'charge', 'sex', 'race', 'dob', 'height',
              'weight', 'eye', 'hair', 'issue_date', ]
    table = re.search((r'<table width="780" border="1" cellpadding="2"
                        ' cellspacing="0" style="border-color:#CCCCCC;
                        ' border-width:1px;" >(.*?)</table>'),
                      page, re.S)
    rows = re.findall('<tr>(.*?)</tr>', table.group(), re.S)
    for row in rows[1:]:
        rowdata = re.findall(r'<td>(.*?)</td>', row)
        rowdata = dict(zip(fields, rowdata))
        yield rowdata
```

Putting it all together

```
def scrape():
    url = 'http://norfolk.gov/Police/wanted_person.asp'
    content = urlopen(url).read()
    urls = re.findall(r'wanted_person_details.asp\?letter=[A-Z]', content)
    urls = ['http://norfolk.gov/Police/%s' % url for url in urls]
    for url in urls:
        letter_content = urlopen(url).read()
        for row in parse_letter_page(letter_content):
            yield row
```

Once you have the data

```
>>> # Save it to a CSV file
```

```
>>> import csv
```

```
>>> writer = csv.DictWriter(  
    open('warrants.csv', 'w'),  
    fields).writerow(rowdata)
```

```
>>> # Save it to a database
```

```
>>> cursor.execute('INSERT INTO warrants  
(%(first)s, %(last)s, %(dob)s, %(charge))',  
rowdata)
```

```
>>> # Or whatever you want
```

Making your life easier

```
>>> from BeautifulSoup import BeautifulSoup
```

- “Beautiful Soup is a Python HTML/XML parser designed for quick turnaround projects like screen-scraping.”

<http://www.crummy.com/software/BeautifulSoup>

Making your life easier

```
>>> # Instead of this:
```

```
>>> table = re.search(r'<table width="780"  
border="1" cellpadding="2" cellspacing="0"  
style="border-color:#CCCCCC;border-width:  
1px;" >(.*)</table>', page, re.S)
```

```
>>> # You can do this:
```

```
>>> table = BeautifulSoup(page).find('table',  
{'width': '780'})
```

Making your life easier

```
>>> # Instead of this:
```

```
>>> rows = re.findall('<tr>(.*?)</tr>',  
table.group(), re.S)
```

```
>>> # You can do this:
```

```
>>> rows = table.findAll('tr')
```

```
>>> # And then this:
```

```
>>> data = [x.renderContents() for x in  
row.findAll('td') for row in rows]
```

Other useful tools

- `httplib2` (<http://code.google.com/p/httplib2>)
“A comprehensive HTTP client library that supports many features left out of other HTTP libraries.” Such as:
 - Caching.
 - Last-modified checking.
 - Compression.

If you're requesting the same pages over and over, you should use `httplib2`.

Other useful tools

- **cookielib**
“The **cookielib** module defines classes for automatic handling of HTTP cookies.”

<http://docs.python.org/library/cookielib.html>

Other useful tools

- **urllib.urlencode(*query*[, *doseq*])**
“Convert a mapping object or a sequence of two-element tuples to a ‘url-encoded’ string, suitable to pass to `urlopen()` ...”

<http://docs.python.org/library/urllib.html>

Other useful tools

- LiveHTTPHeaders
“View HTTP headers of a page and while browsing.”

<https://addons.mozilla.org/en-US/firefox/addon/3829>

These slides will be
available at
<http://bycoffe.com/talks>

Questions

References

- http://norfolk.gov/Police/wanted_person.asp
- <http://docs.python.org/library/urllib2.html>
- <http://docs.python.org/library/re.html>
- <http://www.crummy.com/software/BeautifulSoup>
- <http://code.google.com/p/httpplib2>
- <http://diveintopython3.org/http-web-services.html>
- <http://docs.python.org/library/cookielib.html>
- <http://docs.python.org/library/urllib.html>

Photo Credits

- <http://www.flickr.com/photos/nasmac/531138641/>

Me

- @bycoffe
- bycoffe@gmail.com
- bycoffe.com