BeautifulSoup: Web Scraping with Python

Andrew Peterson

Apr 9, 2013

files available at:
`https://github.com/aristotle-tek/BeautifulSoup_pres`

Roadmap

- Uses: data types, examples...
- Getting Started
- downloading files with wget
- BeautifulSoup: in depth example - election results table
- Additional commands, approaches
- PDFminer
- (time permitting) additional examples

Etiquette/ Ethics

- Similar rules of etiquette apply as Pablo mentioned:
- Limit requests, protect privacy, play nice...

Data/Page formats on the web

- HTML, HTML5 (<!DOCTYPE html>)

```
<div align="center">
        <table width="952" border="0" cellspacing="0" cellpadding="0">
                <tr>
                        <td width="1" bgcolor="cccccc">
                                <img src="../images/transparent.gif" widt
                        </td>
                        <td width="950" background="../images/main_back.p

                                <table width="950" border="0" cellspacing
                                        <tr>
```

Data/Page formats on the web

- HTML, HTML5 (`<!DOCTYPE html>`)
- data formats: XML, JSON
- PDF
- APIs
- other languages of the web: css, java, php, asp.net...
- (don't forget existing datasets)

BeautifulSoup

- General purpose, robust, works with broken tags
- Parses html and xml, including fixing asymmetric tags, etc.
- Returns unicode text strings
- Alternatives: lxml (also parses html), Scrapey
- Faster alternatives: ElementTree, SGMLParser (custom)

Installation

- `pip install beautifulsoup4` or `easy_install beautifulsoup4`
- See: `http://www.crummy.com/software/BeautifulSoup/`
- On installing libraries: `http://docs.python.org/2/install/`

HTML Table basics

`<table>` Defines a table

`<th>` Defines a header cell in a table

`<tr>` Defines a row in a table

`<td>` Defines a cell in a table

HTML Tables

Simple table:

[r1, c1]	[r1, c2]
[r2, c1]	[r2, c2]

HTML Tables

```
<h4>Simple table:</h4>
<table>
<tr>
  <td>[r1, c1] </td>
  <td>[r1, c2] </td>
</tr>
<tr>
  <td>[r2, c1]</td>
  <td>[r2, c2]</td>
</tr>
</table>
```

Example: Election data from html table

Bomi County

Voting Precinct 03001

October 11, 2011 Elections

President/Vice-President Results

Position	PP01
BEYAN, Gladys G. Y. (GDPL)	0
BRUMSKINE, Charles Walker (LP)	1
CHEAPOO, SR., Chea Job (PPP)	0
CHELLEY, James Kpa (OCPOL)	0
FREEMAN, Simeon (MPC)	0
GUSEH, James Sawalla (CUP)	1
JOHNSON, Prince Yormie (NUDP)	2
JOHNSON-SIRLEAF, Ellen (UP)	126
JONES, Marcus Roland (VCP)	0
MASON, Jonathan A. (ULD)	1
MAYSON, Dew Tuan-Wleh (NDC)	0
NDEBE, Manjerngie Cecelia (LRP)	0
SANDY, Kennedy Gbleyah (LTP)	0
TIPOTEH, Togba-Nah (FAPL)	0
TUBMAN, Winston A. (CDC)	10
ZOE, Hananiah (LEP)	0
Total Valid	141
Invalid	3
Total	144

Example: Election data from html table

- election results spread across hundreds of pages
- want to quickly put in useable format (e.g. csv)

Download relevant pages

- website might change at any moment
- ability to replicate research
- limits page requests

Download relevant pages

- I use `wget` (GNU), which can be called from within python
- alternatively `cURL` may be better for macs, or `scrapy`

Download relevant pages

- wget: note the `--no-parent` option!

```
os.system("wget --convert-links --no-clobber \
--wait=4 \
--limit-rate=10K \
-r --no-parent http://www.necliberia.org/results2011/results
```

Step one: view page source

```
<div class="res">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<th>Position</th><th>PP01</th>
<tr><td class="n">BEYAN, Gladys G. Y. (GDPL)</td><td width="35" class="b">0</td></t
<tr><td class="n">BRUMSKINE, Charles Walker (LP)</td><td width="35" class="b">1</td
<tr><td class="n">CHEAPOO, SR., Chea Job (PPP)</td><td width="35" class="b">0</td><
<tr><td class="n">CHELLEY, James Kpa (OCPOL)</td><td width="35" class="b">0</td></t
<tr><td class="n">FREEMAN, Simeon (MPC)</td><td width="35" class="b">0</td></tr>
<tr><td class="n">GUSEH, James Sawalla (CUP)</td><td width="35" class="b">1</td></t
```

Outline of Our Approach

1. identify the county and precinct number
2. get the table:
   - identify the correct table
   - put the rows into a list
   - for each row, identify cells
   - use regular expressions to identify the party & lastname
3. write a row to the csv file

Open a page

- `soup = BeautifulSoup(html_doc)`
- Print all: `print(soup.prettify())`
- Print text: `print(soup.get_text())`

Navigating the Page Structure

- some sites use `div`, others put everything in tables.

find_all

- finds all the Tag and NavigableString objects that match the criteria you give.
- find table rows: `find_all("tr")`
- e.g.:

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

```
</td>
<td width="910">
        <table width="100%" border="0" cellspacing="0" cel
                <tr>
                        <td align="left">
                                <h2>Grand Bassa County</h2
                                <h4>Voting Precinct 09006<
                        </td>
                </tr>
        </table>
        <h3>October 11, 2011 Elections</h3>
        <table width="100%" border="0" cellspacing="0" cel
                <tr>
                        <td align="left" valign="top">
                                <h3>President/Vice-Presid
                                <div class="res">
                                        <table width="100
```

Let's try it out

- We'll run through the code step-by-step

Regular Expressions

- Allow precise and flexible matching of strings
- precise: i.e. character-by-character (including spaces, etc)
- flexible: specify a set of allowable characters, unknown quantities
- `import re`

from xkcd

Regular Expressions: metacharacters

- Metacharacters:

  `| . ^ $ * + ? { } [ ] \ | ( )`

- excape metacharacters with backslash \

Regular Expressions: Character class

- brackets `[ ]` allow matching of any element they contain
- `[A-Z]` matches a capital letter, `[0-9]` matches a number
- `[a-z][0-9]` matches a lowercase letter followed by a number

Regular Expressions: Repeat

- star `*` matches the previous item 0 or more times
- plus `+` matches the previous item 1 or more times
- `[A-Za-z]*` would match only the first 3 chars of `Xpr8r`

Regular Expressions: match anything

- dot `.` will match anything but line break characters `\r` `\n`
- combined with `*` or `+` is very hungry!

Regular Expressions: or, optional

- pipe is for 'or'

  `'abc|123' matches 'abc' or '123' but not 'ab3'`

- question makes the preceeding item optional: `c3?[a-z]+` would match `c3po` and also `cpu`

Regular Expressions: in reverse

- parser starts from beginning of string
- can tell it to start from the end with $

Regular Expressions

- `\d`, `\w` and `\s`
- `\D`, `\W` and `\S` NOT digit (use outside char class)

Regular Expressions

- Now let's see some examples and put this to use to get the party.

Basic functions: Getting headers, titles, body

- `soup.head`
- `soup.title`
- `soup.body`

Basic functions

- `soup.b`
- id: `soup.find_all(id="link2")`
- eliminate from the tree: `decompose()`

Other Methods: Navigating the Parse Tree

- With `parent` you move up the parse tree. With `contents` you move down the tree.
- `contents` is an ordered list of the Tag and NavigableString objects contained within a page element.
- `nextSibling` and `previousSibling`: skip to the next or previous thing on the same level of the parse tree

Data output

- Create simple csv files: `import csv`
- many other possible methods: e.g. use within a `pandas` DataFrame (cf Wes McKinney)

Putting it all together

- Loop over files
- for vote total rows, make the party empty
- print each row with the county and precinct number as columns

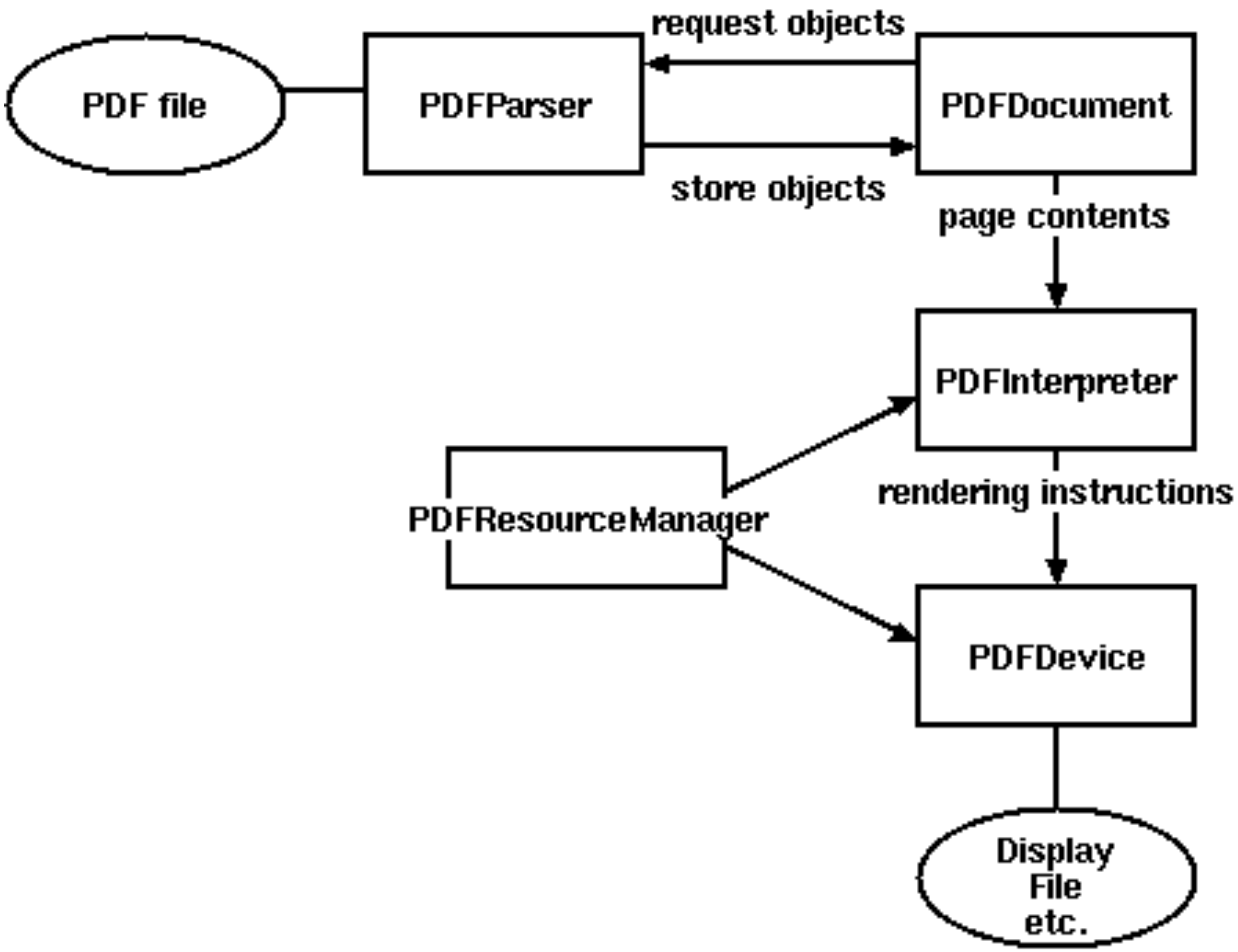
PDFs

- Can extract text, looping over 100s or 1,000s of pdfs.
- not based on character recognition (OCR)

pdfminer

- There are other packages, but `pdfminer` is focused more directly on scraping (rather than creating) pdfs.
- Can be executed in a single command, or step-by-step

pdfminer


request objects
PDF file
PDFParser
PDFDocument
store objects
page contents
PDFInterpreter
PDFResourceManager
rendering instructions
PDFDevice
Display
File
etc.

PDFs

- We'll look at just using it within python in a single command, outputting to a .txt file.
- Sample pdfs from the National Security Archive Iraq War: `http://www.gwu.edu/~nsarchiv/NSAEBB/NSAEBB418/`

Performing an action over all files

- Often useful to do something over all files in a folder.
- One way to do this is with glob:

```
import glob
for filename in glob.glob('/filepath/*.pdf'):
    print filename
```

- see also an example file with pdfminer

Additional Examples

- (time permitting): newspapers, output to pandas...