Extracting data from HTML with Python

I have following text processed by my code in Python:

```
<td>
<a href="http://www.linktosomewhere.net" title="title here">some link</a>
<br />
some data 1<br />
some data 2<br />
some data 3</td>
```

Could you advice me how to extract data from within `<td>` ? My idea is to put it in a CSV file with the following format: `some link, some data 1, some data 2, some data 3` .

I expect that without regular expression it might be hard but truly I still struggle against regular expressions.

I used my code more or less in following manner:

```
tabulka = subpage.find("table")

for row in tabulka.findAll('tr'):
    col = row.findAll('td')
print col[0]
```

and ideally would be to get each td contend in some array. Html above is a result from python.

python html

edited Jun 15 '13 at 19:34
Cairnarvon
11.6k 9 32 51

asked Jun 15 '13 at 18:28
Lormitto
97 1 1 12

You can use Beautifoulsoup crummy.com/software/BeautifulSoup/bs3/documentation.html to extract the information rather then using regular expression. – locojay Jun 15 '13 at 18:32

Don't use regular expressions to parse non-trivial HTML. – Cairnarvon Jun 15 '13 at 19:35

I can use BeautifulSoup to extract data from td but what to do next to work on data inside? – Lormitto Jun 15 '13 at 19:45

2 Answers

Get BeautifulSoup and just use it. It's great.

```
$> easy_install pip
$> pip install BeautifulSoup
$> python
>>> from BeautifulSoup import BeautifulSoup as BS
>>> import urllib2
>>> html = urllib2.urlopen(your_site_here)
```

```
>>> soup = BS(html)
>>> elem = soup.findAll('a', {'title': 'title here'})
>>> elem[0].text
```

edited Jun 15 '13 at 19:14

answered Jun 15 '13 at 18:35

Droogans
3,818 21 48

2 findAll() returns a list or None, and lists and None don't have a .text() method, so your last line is always an error. – 7stud Jun 15 '13 at 18:39

as 7stud stated above it is a problem indeed – Lormitto Jun 15 '13 at 19:03

You shouldn't use regexes on html. You should use BeautifulSoup or lxml. Here are some examples using BeautifulSoup:

Your td tags actually look like this:

```
<td>newline
<a>some link</a>newline
<br />newline
some data 1<br />newline
some data 2<br />newline
some data 3</td>
```

So td.text looks like this:

```
<newline>some link<newline><newline>some data 1<newline>some data 2<newline>some data 3
```

You can see that each string is separated by at least one newline, so that enables you to separate out each string.

```
from bs4 import BeautifulSoup as bs
import re

html = """<td>
<a href="http://www.linktosomewhere.net" title="title here">some link</a>
<br />
some data 1<br />
some data 2<br />
some data 3</td>"""

soup = bs(html)
tds = soup.find_all('td')
csv_data = []

for td in tds:
    inner_text = td.text
    strings = inner_text.split("\n")

    csv_data.extend([string for string in strings if string])

print(",".join(csv_data))

--output:--
some link,some data 1,some data 2,some data 3
```

Or more concisely:

```
for td in tds:
    print(re.sub("\n+", ",", td.text.lstrip() ) )

--output:--
some link,some data 1,some data 2,some data 3
```

But that solution is brittle because it won't work if your html looks like this:

```
<td>
<a href="http://www.linktosomewhere.net" title="title here">some link</a>
<br />some data 1<br />some data 2<br />some data 3</td>
```

Now td.text looks like this:

```
<newline>some link<newline>some data 1some data2some data3
```

And there isn't a way to figure out where some of the strings begin and end. But that just means you can't use td.text--there are still other ways to identify each string:

1)

```
from bs4 import BeautifulSoup as bs
import re
```

```
html = """<td>
<a href="http://www.linktosomewhere.net" title="title here">some link</a>
<br />some data 1<br />some data 2<br />some data 3</td>"""

soup = bs(html)
tds = soup.find_all('td')
csv_data = []

for td in tds:
    a_tags = td.find_all('a')

    for a_tag in a_tags:
        csv_data.append(a_tag.text)
        br_tags = a_tag.findNextSiblings('br')

        for br in br_tags:
            csv_data.append(br.next.strip())  #get the element after the <br> tag

csv_str = ",".join(csv_data)
print(csv_str)

--output:--
some link,some data 1,some data 2,some data 3
```

2)

```
for td in tds:
    a_tag = td.find('a')
    if a_tag: csv_data.append(a_tag.text)

    for string in a_tag.findNextSiblings(text=True):  #find only text nodes
        string = string.strip()
        if string: csv_data.append(string)

csv_str = ",".join(csv_data)
print(csv_str)

--output:--
some link,some data 1,some data 2,some data 3
```

3)

```
for td in tds:
    a_tag = td.find('a')
    if a_tag: csv_data.append(a_tag.text)

    text_strings = a_tag.findNextSiblings( text=re.compile('\S+') )  #find only non-
whitespace text nodes
    csv_data.extend(text_strings)

csv_str = ",".join(csv_data)
print(csv_str)

--output:--
some link,some data 1,some data 2,some data 3
```

edited Jun 23 '13 at 5:56

answered Jun 23 '13 at 1:32

7stud

22.4k 7 41 69