Finishing Regular Expressions & XML / Web Scraping

April 7 2015
Today

• Finish Regular Expressions
• XML Parsing in Python

• Course Calendar
Last Class

• Introduced **iterators** and **match groups**
  – Iterators let you loop through a set of match results without creating a big list upfront
  – Match groups (the parenthesis notation) let you find the text that matches smaller parts of a regex pattern
Last Class

- `re.search('a(\w+)a', 'mechanical')`
- `re.match('m(\.\)', 'mechanical')`
- `re.finditer('\w+', 'This is a text.')`

Match 1 (this)
Match 2 (is)
Match 3 (a)
Match 4 (text)

Match
- `group(0)` Matched string (in red)
- `start(0)` or `end(0)` Start or Finishing position
- `group(1)` Matched 1st parenthesis group (in **bold**)

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences
## Data Structures

### Lists

<table>
<thead>
<tr>
<th>indices</th>
<th>'a'</th>
<th>'b'</th>
<th>'c'</th>
<th>'d'</th>
<th>'e'</th>
<th>'f'</th>
<th>'g'</th>
<th>'h'</th>
<th>'i'</th>
<th>'j'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

### Dictionaries

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'alice'</td>
<td>'401-111-1111'</td>
</tr>
<tr>
<td>'carol'</td>
<td>'401-222-2222'</td>
</tr>
<tr>
<td>'bob'</td>
<td>'401-333-3333'</td>
</tr>
</tbody>
</table>

### Iterators

- Match 1
- Match 2
- Match 3
- Match 4
- Matched String
- Matched String Start
- Matched String End
Last Class

• Introduced **iterators** and **match groups**
  – Iterators let you loop through a set of match results without creating a big list upfront
  – Match groups (the parenthesis notation) let you find the text that matches smaller parts of a regex pattern

• *Almost* finished ACT 3-2 in class
Substitution using RegEx in Python

• `re.sub(pattern, replacement, string)`
  • Returns the *modified string*...
  • ... with all occurrences of *pattern* substituted by *replacement*

• *Ex.::*
  • `re.sub(’\d’, ’’, ’123foo’)`
    • Returns ‘foo’
  • `re.sub(’\w+’, ’<word>’, ’This is a text.’)`
    • Returns ’<word> <word> <word> <word>‘
  • `re.sub(’\n’, ’ – ’, ’Line 1\nLine 2’)`
    • Returns ’Line 1 – Line 2'
Splitting using RegEx in Python

- `re.split(pattern, string)`
  - Returns a *list of words*...
  - ... separated by occurrences of ...

- **Ex.:**
  - `re.split('\s', 'Text with whitespaces.')`
    - Returns `['Text', 'with', 'whitespaces.']`
  - `re.split('\n', 'Line 1\nLine 2')`
    - Returns `['Line 1', 'Line 2']`

In groups, do Task 5
Regex exercises

• Let’s try to create the following expressions:

• Capitalized words
• Hyphenated names
• Dates like 10/13/2014 or 4/4/03
• Words in quotes

In groups, do Task 6
Web Scraping Introduction

• Why isn’t there a nice “importXML” in Python?

• *lxml* module
  – Not in the default installation
  – Install with “easy_install lxml” in your terminal (Mac OS X terminal or Windows console)
    • That’s how you install any other non-default Python module
Web Scraping Introduction

Open the *Terminal* program and type this (example for Mac OS X):
import lxml.etree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print subnode.tag, ': ', subnode.text
    for subnode in node.xpath('./color'):
        print subnode.tag, ': ', subnode.text

contents:
<inventory>
    <car>
        <year> 2010 </year>
        <color> black </color>
    </car>
    <car>
        <year> 2012 </year>
        <color> red </color>
    </car>
    <car>
        <year> 2014 </year>
        <color> yellow </color>
    </car>
</inventory>
Using lxml

```python
import lxml.etree as et

filename = 'example.xml'
file = open(filename, 'r')
contents = file.read()
file.close()

contents:

<inventory>
    <car>
        <year> 2010 </year>
        <color> black </color>
    </car>
    <car>
        <year> 2012 </year>
        <color> red </color>
    </car>
    <car>
        <year> 2014 </year>
        <color> yellow </color>
    </car>
</inventory>

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print subnode.tag, ': ', subnode.text
    for subnode in node.xpath('./color'):
        print subnode.tag, ': ', subnode.text
```

Internal rep. of the XML

Using lxml

```
import lxml.etree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print subnode.tag, ': ', subnode.text
    for subnode in node.xpath('./color'):
        print subnode.tag, ': ', subnode.text
```

**contents:**

```
<inventory>
 <car>
  <year> 2010 </year>
  <color> black </color>
 </car>
 <car>
  <year> 2012 </year>
  <color> red </color>
 </car>
 <car>
  <year> 2014 </year>
  <color> yellow </color>
 </car>
</inventory>
```
Using lxml

```python
import lxml.etree as et
filename = 'example.xml'
file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print(subnode.tag, ': ', subnode.text)
    for subnode in node.xpath('./color'):
        print(subnode.tag, ': ', subnode.text)
```

**contents:**

```
<inventory>
  <car>
    <year>2010</year>
    <color>black</color>
  </car>
  <car>
    <year>2012</year>
    <color>red</color>
  </car>
  <car>
    <year>2014</year>
    <color>yellow</color>
  </car>
</inventory>
```
Using lxml

```python
import lxml.etree as et
filename = 'example.xml'
file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print(subnode.tag, ': ', subnode.text)
    for subnode in node.xpath('./color'):
        print(subnode.tag, ': ', subnode.text)
```

Contents:

```xml
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```
Using lxml

```python
import lxml.etree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print(subnode.tag, ': ', subnode.text)
    for subnode in node.xpath('./color'):
        print(subnode.tag, ': ', subnode.text)
```

contents:

```xml
<inventory>
  
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>

</inventory>
```

`subnode.text` gives back text
`subnode.tag` gives back tag
Using lxml

```python
import lxml.etree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)

for node in tree.xpath('//car'):
    for subnode in node.xpath('./year'):
        print subnode.tag, ': ', subnode.text
    for subnode in node.xpath('./color'):
        print subnode.tag, ': ', subnode.text

node.attrib gives back attribute dictionary:
{‘c’: ‘good’, ‘h’:’10’}
```

contents:

```
<inventory>
  <car c='good' h='10'>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```
Getting data from the web

1) Use urllib

url = 'http://www.example.com/cars.xml'

remoteFile = urllib.urlopen(url)
contents = remoteFile.read()
remoteFile.close()

2) Use lxml

for node in tree.xpath('//car'):
    print node.text
    print node.tag
    attributes = node.attrib
    for key in attributes.keys:
        print key, attributes[key]
Show me how that’s useful!

• Go to:
  – https://maps.googleapis.com/maps/api/geocode/xml?address=115+Waterman+Street,+Providence,+RI

• This is Google letting you:
  – Give an address
  – Get back an XML containing latitude/longitude

```xml
<location>
  <lat>41.8271609</lat>
  <lng>-71.3995390</lng>
</location>
```
Show me how that’s useful!

• Go to:

• This is Google letting you:
  – Give latitude/longitude
  – Get back an XML with address information
This happens a lot!

• Web services *expose functionality* via *XML*
• Or via something called *JSON*
  – Even *more convenient*!

• Stay tuned!
• Next class we *interact with Google Maps!*
# Tentative Schedule

## Regex and Other Tools

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Material</th>
<th>HW</th>
<th>Date</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 3/31</td>
<td>Regular Expressions</td>
<td>[slides]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ACT 3-1</strong></td>
<td></td>
<td></td>
<td><strong>PythonRE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ACT3-1.py poem.txt regexpal.com</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 4/2</td>
<td>More Regular Expressions</td>
<td>[slides]</td>
<td></td>
<td>T 4/7</td>
<td><strong>HW3-2.py casey.txt</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ACT 3-2 Plain Text Dictionary</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Dict. Words</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Dict. Word Fetcher</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>RE Tester</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>HW3-2 (3-1 was cancelled)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 4/7</td>
<td>Finishing Regular Expressions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 4/9</td>
<td>Google Earth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 4/14</td>
<td>Twitter and While Loops</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Putting It All Together

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>(Proposal)</th>
<th>(Project)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Th 4/16</td>
<td>Final Project</td>
<td></td>
<td>M 5/4</td>
</tr>
<tr>
<td>T 4/21</td>
<td>Project Workshop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 4/23</td>
<td>Python Odds and Ends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 4/28</td>
<td>Project Workshop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 4/30</td>
<td>Rising Computing Applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 5/5</td>
<td>Last Class Day!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Tentative Schedule

## Regex and Other Tools

<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 3/31</td>
<td>Regular Expressions [slides]</td>
<td>ACT 3-1 ACT3-1.py poem.txt regexpal.com</td>
</tr>
<tr>
<td>Th 4/2</td>
<td>More Regular Expressions [slides]</td>
<td>ACT 3-2 Plain Text Dictionary Dict. Words Dict. Word Fetcher RE Tester</td>
</tr>
<tr>
<td>T 4/7</td>
<td>Finishing Regular Expressions</td>
<td></td>
</tr>
<tr>
<td>Th 4/9</td>
<td>Google Earth</td>
<td></td>
</tr>
<tr>
<td>T 4/14</td>
<td>Twitter and While Loops</td>
<td></td>
</tr>
</tbody>
</table>

## Putting It All Together

<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Th 4/16</td>
<td>Final Project</td>
<td></td>
</tr>
<tr>
<td>T 4/21</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/23</td>
<td>Python Odds and Ends</td>
<td></td>
</tr>
<tr>
<td>T 4/28</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/30</td>
<td>Rising Computing Applications</td>
<td></td>
</tr>
<tr>
<td>T 5/5</td>
<td>Last Class Day!</td>
<td></td>
</tr>
</tbody>
</table>
# Tentative Schedule

## Regex and Other Tools

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 3/31</td>
<td>Regular Expressions [slides]</td>
<td><a href="#">ACT 3-1</a> <a href="#">ACT3-1.py</a> poem.txt <a href="#">regexpal.com</a></td>
<td>PythonRE</td>
</tr>
<tr>
<td>Th 4/2</td>
<td>More Regular Expressions [slides]</td>
<td><a href="#">ACT 3-2</a> <a href="#">Plain Text Dictionary</a> Dict. Words Dict. Word Fetcher RE Tester</td>
<td>HW3-2 (3-1 was cancelled)</td>
</tr>
<tr>
<td>T 4/7</td>
<td>Finishing Regular Expressions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 4/9</td>
<td>Google Earth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 4/14</td>
<td>Twitter and While Loops</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Putting It All Together

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Th 4/16</td>
<td>Final Project</td>
<td>(Proposal) T 4/21 (Project) M 5/4</td>
</tr>
<tr>
<td>T 4/21</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/23</td>
<td>Python Odds and Ends</td>
<td></td>
</tr>
<tr>
<td>T 4/28</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/30</td>
<td>Rising Computing Applications</td>
<td></td>
</tr>
<tr>
<td>T 5/5</td>
<td>Last Class Day!</td>
<td></td>
</tr>
</tbody>
</table>

[ACT 3-1](#) [ACT3-1.py](#) poem.txt [regexpal.com](#) [HW3-2.py](#) casey.txt
# Tentative Schedule

## Regex and Other Tools

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 3/31</td>
<td>Regular Expressions [slides]</td>
<td>ACT 3-1, ACT3-1.py, poem.txt, regexpal.com</td>
</tr>
<tr>
<td>Th 4/2</td>
<td>More Regular Expressions [slides]</td>
<td>ACT 3-2 Plain Text Dictionary, Dict. Words, Dict. Word Fetcher, RE Tester, HW3-2 (3-1 was cancelled)</td>
</tr>
<tr>
<td>T 4/7</td>
<td>Finishing Regular Expressions</td>
<td></td>
</tr>
<tr>
<td>Th 4/9</td>
<td>Google Earth</td>
<td></td>
</tr>
<tr>
<td>T 4/14</td>
<td>Twitter and While Loops</td>
<td></td>
</tr>
</tbody>
</table>

## Putting It All Together

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Th 4/16</td>
<td>Final Project</td>
<td>(Proposal) T 4/21 (Project) M 5/4</td>
</tr>
<tr>
<td>T 4/21</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/23</td>
<td>Python Odds and Ends</td>
<td></td>
</tr>
<tr>
<td>T 4/28</td>
<td>Project Workshop</td>
<td></td>
</tr>
<tr>
<td>Th 4/30</td>
<td>Rising Computing Applications</td>
<td></td>
</tr>
<tr>
<td>T 5/5</td>
<td>Last Class Day!</td>
<td></td>
</tr>
</tbody>
</table>