Ch10-2-Files-Advanced

August 7, 2020

1 Advanced Topics on Files

1.1 Working with HTML files

- fetch an HTML page from web
- parse the HTML file with BeautifulSoup library

[1]:
```
# fetch an html page
import urllib.request
url = 'https://rambasnet.github.io/teaching.html'
localfile = 'teaching.html'
urllib.request.urlretrieve(url, localfile)
```

[1]:
```
('teaching.html', <http.client.HTTPMessage at 0x111004828>)
```

[2]:
```
with open(localfile) as f:
    data = f.read()
words = data.split(' ')
print('There are {0} words in the file.'.format(len(words)))
```

```
There are 10165 words in the file.
```

1.2 parsing HTML using BeautifulSoup library

- install Beautifulsoup library $ pip install bs4
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/#
- Alternative is nltk (Natural Language Toolkit) library
- http://www.nltk.org/

[3]:
```
# can run terminal/bash commands from notebook using !
! pip install bs4
```

```
Requirement already satisfied: bs4 in
/Users/rbasnet/miniconda3/lib/python3.7/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in
/Users/rbasnet/miniconda3/lib/python3.7/site-packages (from bs4) (4.7.1)
Requirement already satisfied: soupsieve>=1.2 in
/Users/rbasnet/miniconda3/lib/python3.7/site-packages (from beautifulsoup4->bs4)
(1.9.1)
```

[4]:
```
from bs4 import BeautifulSoup
localfile = 'teaching.html'
with open(localfile) as f:
    soup = BeautifulSoup(f.read(), 'lxml')
text = soup.get_text()
print(text)
```

```
Ram Basnet | Homepage




Dr. Ram Basnet
Associate Professor of Computer Science









Home


Teaching


Research


Resources


Contact
```

Teaching

Teaching Interests

Cyber Security

Python, C++, Java, Database, JavaScript

Data Science

Web Design and Secure Web App Development

Courses Taught at CMU

CSCI 106 - Web Page I
6

CSCI 110 - Beg. Prog. Python & Lab
6

CS1 - Foundation of CS
7

CS2 - Data Structures
7

CSCI 206 - Web Page II
2

CS3 - Intro to Algorithms
2

CSCI 310 - Adv. Prog. Python
7

CSCI 420 - Cyber Security
5

CSCI 465 - Net/App Security
5

CURRENT SCHEDULE

Mon
Tues
Wed
Thrs
Fri

8:00

CS0WS 120

CS0-LWS 120

CS0WS 120

CS0-LWS 120

CS0WS 120

8:30

9:00

Ad PyWS 118

Off. Hr.CH 321

Ad PyWS 118

Off. Hr.CH 321

Off. Hr.WS 116 (CRL)

9:30

10:00

Off. Hr.CH 321

10:30

11:00

Net/App SecWS 205

Net/App SecWS 205

Net/App SecWS 205

11:30

12:00

12:30

1:00

CS 3CH 276

CS 3CH 276

CS 3CH 276

1:30

2:00

Off. Hr.CH 321

2:30

3:00

3:30

```
Home | Teaching |
        Research  |
        Resources |
         Contact       ©
        2019
```

```
    var dt = new Date()
    document.getElementById("year").innerHTML = dt.getFullYear()
    var windowSize = window.matchMedia("(max-width: 375px)")
    if (windowSize.matches) {
        var element = document.getElementById("cmu-logo")
        element.setAttribute("style", "visibility: hidden;")
    }


    /*
    var navul = document.getElementById("navul")
    var alists = navul.getElementsByClassName("nav-link")
    for (var i = 0; i < alists.length; i++) {
        alists[i].addEventListener("click", function() {
            var current = document.getElementsByClassName("active")
            current[0].className = current[0].className.replace(" active", "")
            this.className += " active"
        })
    }
    */
    var hrefString = document.location.href
        ? document.location.href
        : document.location
    var url = hrefString.split("/") //replace string with location.href
    var navLinks = document
```

```
    .getElementById("navul")
    .getElementsByClassName("nav-item")
var currentPage = url[url.length - 1]
for (var i = 0; i < navLinks.length; i++) {
    var link = navLinks[i].getElementsByClassName("nav-link")[0]
    var lb = link.href.split("/")
    if (lb[lb.length - 1] == currentPage) {
        navLinks[i].className += " active"
    } else {
        navLinks[i].className = navLinks[i].className.replace(" active", "")
    }
}




;(function(i, s, o, g, r, a, m) {
    i["GoogleAnalyticsObject"] = r
    ;(i[r] =
        i[r] ||
        function() {
            ;(i[r].q = i[r].q || []).push(arguments)
        }),
        (i[r].l = 1 * new Date())
    ;(a = s.createElement(o)), (m = s.getElementsByTagName(o)[0])
    a.async = 1
    a.src = g
    m.parentNode.insertBefore(a, m)
})(
    window,
    document,
    "script",
    "//www.google-analytics.com/analytics.js",
    "ga"
)

ga("create", "UA-46738331-1", "coloradomesa.edu")
ga("send", "pageview")
```

[5]:
```
# break into lines and remove leading and trailing space on each line
lines = [line.strip() for line in text.splitlines()]
```

[6]:
```
print(lines[:20])
```

```
['', '', 'Ram Basnet | Homepage', '', '', '', '', '', '', '', '', '', '', '',
'', 'Dr. Ram Basnet', 'Associate Professor of Computer Science', '', '', '']
```

[16]:
```
# create list of words by spliting multi-word elements
words = [word.strip().lower() for line in lines for word in line.split()]
```

[17]:
```
print(words[:20])
```

```
['ram', 'basnet', '|', 'homepage', 'dr.', 'ram', 'basnet', 'associate',
'professor', 'of', 'computer', 'science', 'home', 'teaching', 'research',
'resources', 'contact', 'teaching', 'teaching', 'interests']
```

[9]:
```
print('There are {0} words in the file.'.format(len(words)))
```

```
There are 367 words in the file.
```

1.3 Find histogram of words

- use DefaultDict found in collections module
- https://docs.python.org/3/library/collections.html

[18]:
```
from collections import defaultdict
```

[28]:
```
hist = defaultdict(int)
for w in words:
    hist[w] += 1
```

[29]:
```
# print top 10 most common words
listHist = [(k, v) for k, v in hist.items()]
```

[31]:
```
print(listHist[:10])
```

```
[('ram', 2), ('basnet', 2), ('|', 5), ('homepage', 1), ('dr.', 1), ('associate',
1), ('professor', 1), ('of', 2), ('computer', 1), ('science', 2)]
```

[34]:
```
listHist.sort(key = lambda x: x[1], reverse=True)
```

[35]:
```
print(listHist[:10])
```

```
[('=', 25), ('var', 12), ('-', 11), ('{', 8), ('csci', 6), ('|', 5), ('i', 5),
('120', 5), ('off.', 5), ('}', 5)]
```

1.4 working with binary files

- the following example copies a binary file such as image

[11]:
```
fileSrc = './resources/brain.jpg'
fileDst = 'brain-copy.jpg'
with open(fileSrc, 'rb') as rbf:
    #rb - read binary mode
    data = rbf.read() # read the whole binary file
    with open(fileDst, 'wb') as wbf:
        wbf.write(data) # write the whole binary file
```

1.5 use checksum to compare if two files match exactly!

- checksum makes sure that not a single bit is different between the two files
- used in security
- import and use hashlib - https://docs.python.org/3/library/hashlib.html

[15]:
```
import hashlib
file1Contents = open(fileSrc, 'rb').read()
file2Contents = open(fileDst, 'rb').read()

file1ChkSum = hashlib.sha256(file1Contents).hexdigest()
file2ChkSum = hashlib.sha256(file2Contents).hexdigest()
if (file1ChkSum == file2ChkSum):
    print('two files checksums match!')
else:
    print('oops! two files checksums do NOT match!')
```

```
two files checksums match!
```

1.6 Python object serialization with pickle library

- https://docs.python.org/3/library/pickle.html
- pickle module implements binary protocols for serializing and de-serializing a Python object
- Pickling - serializing python object
- Un-pickling - de-serializing python object (inverse operation)

[38]:
```
import pickle
alist = list(range(2, 21, 2))
```

[37]:
```
print(alist)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

[40]:
```
# lets pickle alist
pickleFile = 'myPickle.pkl'
with open(pickleFile, 'wb') as p:
    pickle.dump(alist, p)
```

[41]:
```
# lets unpickle alist
with open(pickleFile, 'rb') as p:
```

```
    blist = pickle.load(p)
```

[42]:
```
alist == blist
```

[42]:
```
True
```

[ ]:
```

```