Ch10-2-Files-Advanced

September 22, 2021

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 0x7fb266020190>)
```

[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 528 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/

1.3 Installing Parsers

- supports the HTML parser included in Python's standard library
- also supports a number of third-party Python parsers such as very fast `lxml` parser

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

```
Requirement already satisfied: bs4 in
/Users/rbasnet/miniconda3/lib/python3.8/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in
/Users/rbasnet/miniconda3/lib/python3.8/site-packages (from bs4) (4.9.3)
Requirement already satisfied: soupsieve>1.2 in
/Users/rbasnet/miniconda3/lib/python3.8/site-packages (from beautifulsoup4->bs4)
(2.0.1)
```

[4]:
```
# install lxml parser
! pip install lxml
```

```
Requirement already satisfied: lxml in
/Users/rbasnet/miniconda3/lib/python3.8/site-packages (4.6.1)
```

[5]:
```
from bs4 import BeautifulSoup
localfile = 'teaching.html'
with open(localfile) as f:
    #soup = BeautifulSoup(f, 'lxml') # used to but now not working!
    soup = BeautifulSoup(f, 'html.parser')
text = soup.get_text()
print(text)
```

```
Dr. Ram BasnetAssociate Professor of Computer
ScienceHomeTeachingResearchResourcesContactTeachingTeaching
InterestsCybersecurityPython, C++, JavaScript, DatabasesData Science and ML
ApplicationsWeb Design and Secure Web App DevelopmentCourses Taught at CMUCSCI
106- Web Design I  6CSCI 100- Beg. Python  8CSCI 111- CS1  10CSCI 112- CS2
 7CSCI 206- Web2  2CSCI 250- CS3  3CSCI 310- Adv. Python  9CSCI 310- Adv.
JavaScript  2CSCI 370- Computer Security  5CSCI 420- Cybersecurity  7CSCI 465-
Net/App Security  7CSCI 396- Machine Learning  1FALL 2021
SCHEDULETimeMonTuesWedThrsFri9:00Off. Hr.CH 3299-9:50Off. Hr.CH 3299-9:50Off.
Hr.CH 3299-9:50Off. Hr.CH 3299-9:50Off. Hr.CH
3299-9:509:3010:00     10:30     11:00Net/App SecCH 31511-11:50 Net/App SecCH
31511-11:50 Net/App SecCH 31511-11:5011:30  12:00     12:30     1:00CS1WS
1201-1:50CS1WS 1201-1:50CS1WS 1201-1:50CS1WS 1201-1:50CS1WS
1201-1:501:302:00     2:30     3:00 PythonCH 2763-3:50 PythonCH
2763-3:50 3:30   4:00     | Home | Teaching | Research | Resources | Contact |
 © 2021
```

[6]:
```
# break into lines and remove leading and trailing space on each line
lines = [line.strip() for line in text.splitlines()]
```

[7]:
```
print(lines[:20])
```

```
['Dr. Ram BasnetAssociate Professor of Computer
ScienceHomeTeachingResearchResourcesContactTeachingTeaching
InterestsCybersecurityPython, C++, JavaScript, DatabasesData Science and ML
ApplicationsWeb Design and Secure Web App DevelopmentCourses Taught at CMUCSCI
```

```
106- Web Design I \xa06CSCI 100- Beg. Python \xa08CSCI 111- CS1 \xa010CSCI 112-
CS2 \xa07CSCI 206- Web2 \xa02CSCI 250- CS3 \xa03CSCI 310- Adv. Python \xa09CSCI
310- Adv. JavaScript \xa02CSCI 370- Computer Security \xa05CSCI 420-
Cybersecurity \xa07CSCI 465- Net/App Security \xa07CSCI 396- Machine Learning
\xa01FALL 2021 SCHEDULETimeMonTuesWedThrsFri9:00Off. Hr.CH 3299-9:50Off. Hr.CH
3299-9:50Off. Hr.CH 3299-9:50Off. Hr.CH 3299-9:50Off. Hr.CH
3299-9:509:3010:00\xa0\xa0\xa0\xa0\xa010:30\xa0\xa0\xa0\xa0\xa011:00Net/App
SecCH 31511-11:50\xa0Net/App SecCH 31511-11:50\xa0Net/App SecCH 31511-11:5011:30
\xa0\xa012:00\xa0\xa0\xa0\xa0\xa012:30\xa0\xa0\xa0\xa0\xa01:00CS1WS
1201-1:50CS1WS 1201-1:50CS1WS 1201-1:50CS1WS 1201-1:50CS1WS
1201-1:501:302:00\xa0\xa0\xa0\xa0\xa02:30\xa0\xa0\xa0\xa0\xa03:00\xa0PythonCH
2763-3:50\xa0PythonCH 2763-3:50\xa03:30\xa0\xa0\xa04:00\xa0\xa0\xa0\xa0\xa0|\xa0
Home\xa0|\xa0Teaching\xa0|\xa0Research\xa0|\xa0Resources\xa0|\xa0Contact\xa0|
\xa0\xa0 © 2021']
```

[8]:
```
# create list of words by spliting multi-word elements
words = [word.strip().lower() for line in lines for word in line.split()]
```

[9]:
```
print(words[:20])
```

```
['dr.', 'ram', 'basnetassociate', 'professor', 'of', 'computer',
'sciencehometeachingresearchresourcescontactteachingteaching',
'interestscybersecuritypython,', 'c++,', 'javascript,', 'databasesdata',
'science', 'and', 'ml', 'applicationsweb', 'design', 'and', 'secure', 'web',
'app']
```

[10]:
```
print('There are {0} words in the file.'.format(len(words)))
```

```
There are 119 words in the file.
```

1.4 Find histogram of words

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

[11]:
```
from collections import defaultdict
```

[12]:
```
hist = defaultdict(int)
for w in words:
    hist[w] += 1
```

[13]:
```
# convert dict into a list of tuples
listHist = [(k, v) for k, v in hist.items()]
```

[14]:
```
# print first 10 words and their frequency
print(listHist[:10])
```

```
[('dr.', 1), ('ram', 1), ('basnetassociate', 1), ('professor', 1), ('of', 1),
```

```
('computer', 2), ('sciencehometeachingresearchresourcescontactteachingteaching',
1), ('interestscybersecuritypython,', 1), ('c++,', 1), ('javascript,', 1)]
```

[15]:
```
# sort list based on frequency in reverse order
listHist.sort(key = lambda x: x[1], reverse=True)
```

[16]:
```
# print the top 10 most frequent words
print(listHist[:10])
```

```
[('|', 6), ('hr.ch', 5), ('3299-9:50off.', 4), ('1201-1:50cs1ws', 4), ('7csci',
3), ('net/app', 3), ('secch', 3), ('computer', 2), ('and', 2), ('design', 2)]
```

1.4.1 Use Counter collection

- easier way!

[17]:
```
from collections import Counter
```

[18]:
```
hist = Counter(words)
```

[19]:
```
hist.most_common(10)
```

[19]:
```
[('|', 6),
 ('hr.ch', 5),
 ('3299-9:50off.', 4),
 ('1201-1:50cs1ws', 4),
 ('7csci', 3),
 ('net/app', 3),
 ('secch', 3),
 ('computer', 2),
 ('and', 2),
 ('design', 2)]
```

1.5 working with binary files

- the following example copies a binary file such as image

[20]:
```
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
```

1.6 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

[21]:
```
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!
```

1.7 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
- Unpickling - deserializing python object (inverse operation)
- Unpickling untrusted picked files could have security implications
  - e.g., executing system commands; installing and executing third-party malicious packages and modules; etc.
  - for more: https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization

[22]:
```
import pickle
alist = list(range(2, 21, 2))
```

[23]:
```
print(alist)
```

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

[24]:
```
# let's pickle alist; serialize a list
pickleFile = 'myPickle.pkl'
with open(pickleFile, 'wb') as p:
    pickle.dump(alist, p)
```

[25]:
```
# lets unpickle alist; deserialize a list
with open(pickleFile, 'rb') as p:
    blist = pickle.load(p)
```

[26]:
```
alist == blist
```

[26]:
```
True
```

[27]:
```
# dump Counter
with open('wordCounter.pkl', 'wb') as p:
    pickle.dump(hist, p)
```

[28]:
```
# load pickle
with open('wordCounter.pkl', 'rb') as p:
    newHist = pickle.load(p)
```

[29]:
```
hist == newHist
```

[29]:
```
True
```

[30]:
```
newHist.most_common(3)
```

[30]:
```
[('|', 6), ('hr.ch', 5), ('3299-9:50off.', 4)]
```

[ ]:
```

```