OPEN SOURCE SOFTWARE LAB (15B17CI575)

Lab Assignment 8 (Practice Lab)

Topic Coverage: Python- BeautifulSoup, MongoDb

"Web scraping (web harvesting or web data extraction) is a computer software technique of extracting information from websites."

- How to install beautifulsoup library and pip using requests in python

  i. `$ pip install requests`

  ii. `$ pip install beautifulsoup4`

- How to install beautifulsoup library using setup.py in python

  iii. `$ python setup.py install`

  - How to create a soup using HTML parser

  iv.
```
from bs4 import BeautifulSoup
soup = BeautifulSoup("<html><p>This is <b>invalid HTML</p></html>", "html.parser")
```

  - Extracting URL's from any website

  v.
```
from bs4 import BeautifulSoup
import requests
url = raw_input("Enter a website to extract the URL's from: ")
r  = requests.get("http://" +url)
data = r.text
soup = BeautifulSoup(data)
for link in soup.find_all('a')
    print(link.get('href'))
```

  - The BeautifulSoup object can accept two arguments. The first argument is the actual markup, and the second argument is the parser that you want to use. The different parsers are:

html.parser, lxml, and html5lib. The lxml parser has two versions, an HTML parser and an XML parser.

- Extracting Title, Headings, and Links of a website

vi.
```
import requests
from bs4 import BeautifulSoup
req = requests.get('https://en.wikipedia.org/wiki/Python_(programming_language)')
soup = BeautifulSoup(req.text, "lxml")
print(soup.title)
print(soup.title.name)
print(soup.title.string)
```

- Extracting the main heading or the first paragraph

vii.
```
print(soup.h1)
print(soup.h1.string)
```

viii.
```
soup.h1['class'] = 'firstHeading, mainHeading'
soup.h1.string.replace_with("Python - Programming Language")
del soup.h1['lang']
del soup.h1['id']
```

Web Crawling using pyMongo and storing data in MongoDb

- Parsing data with Beautiful soup

```
import requests
from bs4 import BeautifulSoup
req = requests.get('http://www.usamega.com/mega-millions-history.asp?p=1')
soup = BeautifulSoup(req.text)
print  soup('table')[4].findAll('tr')[1].findAll('td')[1].a.string
print  soup('table')[4].findAll('tr')[1].findAll('td')[3].b.string
print soup('table')[4].findAll('tr')[1].findAll('td')[3].strong.string
```

Final Script With Mongodb Integration

```
import urllib2

from bs4 import BeautifulSoup

from pymongo import Connection

host = 'localhost'
```

```
database  =  'lotto'

collection = 'mega_millions'

def mongo_connection():

   con = Connection(host)

   col = con[database][collection]

   return col

def main():

   col = mongo_connection()

   page_num = 1

   total_pages = 63

   while True:

      if page_num > total_pages: break

         page_num = str(page_num)

      soup = BeautifulSoup(urllib2.urlopen('http://www.usamega.com/mega-millions-
history.asp?p='+page_num).read())

      for row in soup('table')[4].findAll('tr'):

         win_dict = {}

         tds = row('td')

         if tds[1].a is not None:

            win_dict['date'] = tds[1].a.string

            if tds[3].b is not None:

               num_list = []

               #Told you we would get back to it

               number_list = tds[3].b.string.split('&middot;')

               for num in number_list:

                  num_list.append(int(num))
            win_dict['numbers'] = num_list

            mega_number = tds[3].strong.string

            win_dict['mega_number'] = int(mega_number)

            col.insert(win_dict)

        page_num = int(page_num)

        page_num += 1

if __name__== "__main__":

    main()
```

- Refer to the link http://www.briancarpio.com/2012/12/02/website-scraping-with-python-and-beautiful-soup/

Sample code 1: (Structuring data using pandas library)

Refer to the link: https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/

```
import urllib2

import pandas as pd

wiki = "https://en.wikipedia.org/wiki/List_of_state_and_union_territory_capitals_in_India"

page = urllib2.urlopen(wiki)

from bs4 import BeautifulSoup

soup = BeautifulSoup(page)

all_tables=soup.find_all('table')

print(all_tables)

right_table=soup.find('table', class_='wikitable sortable plainrowheaders')
```

```
print(right_table)

A=[]

B=[]

C=[]

D=[]

E=[]

F=[]

G=[]

for row in right_table.findAll("tr"):

    cells = row.findAll('td')

    states=row.findAll('th') #To store second column data

    if len(cells)==6: #Only extract table body not heading

        A.append(cells[0].find(text=True))

        B.append(states[0].find(text=True))

        C.append(cells[1].find(text=True))

        D.append(cells[2].find(text=True))

        E.append(cells[3].find(text=True))

        F.append(cells[4].find(text=True))

        G.append(cells[5].find(text=True))

df=pd.DataFrame(A,columns=['Number'])

df['State/UT']=B

df['Admin_Capital']=C

df['Legislative_Capital']=D

df['Judiciary_Capital']=E

df['Year_Capital']=F
```

df['Former_Capital']=G

print(df)

Processing data stored in MongoDb using Pandas

Create a MongoClient

First, import things we will need. Use pymongo to connect to the "test" database. Specify that we want to use the collection "people" in this database.
In [1]:

```
import os
import pandas as pd
import numpy as np
from IPython.core.display import display, HTML
import pymongo
from pymongo import MongoClient
print 'Mongo version', pymongo._version
client = MongoClient('localhost', 27017)
db = client.test
collection = db.people
```

```
Mongo version 3.3.0
```

Import data into the database

Import data from a json file into the MongoDB database "test", collection "people". We can do this using the insert method, but for simplicity we execute a "mongoimport" in a shell environment, but first we drop the collection if it already exists.
In [2]:

```
collection.drop()
os.system('mongoimport -d test -c people dummyData.json')
```

Out[2]:

```
0
```

Check if you can access the data from the MongoDB.

We use find() to get a cursor to the documents in the data. Let's see who the three youngest persons in this data are. Sort the results by the field "Age", and print out the first three documents. Note the structure of documents, it is the same as the documents we imported from the json file, but it has unique values for the new "_id" field.

In [3]:

```
cursor = collection.find().sort('Age',pymongo.ASCENDING).limit(3)
for doc in cursor:
        print doc
```

```
{u'Country': u'Serbia', u'Age': 18, u'_id': ObjectId('58d690f11ac4479b459dfdf6'), u'Name': u'Sawyer, Neve M.',
u'Location': u'-34.37446, 174.0838'}

{u'Country': u'Somalia', u'Age': 19, u'_id': ObjectId('58d690f11ac4479b459dfdbc'), u'Name': u'Townsend, Cadman I.',
u'Location': u'-87.69188, -144.16138'}

{u'Country': u'Eritrea', u'Age': 20, u'_id': ObjectId('58d690f11ac4479b459dfdde'), u'Name': u'Graham, Emerald O.',
u'Location': u'61.35398, 28.04381'}
```

Aggregation in MongoDB

Here is a small demonstration of the aggregation framework. We want to create a table of the number of persons in each country and their average age. To do it we group by country. We extract the results from MongoDB aggregation into a pandas dataframe, and use the country as index.

In [4]:

```
pipeline = [
                {"$group": {"_id":"$Country",
                          "AvgAge":{"$avg":"$Age"},
                          "Count":{"$sum":1},
                }},
                {"$sort":{"Count":-1,"AvgAge":1}}
]
aggResult = collection.aggregate(pipeline) # returns a cursor
df1 = pd.DataFrame(list(aggResult)) # use list to turn the cursor to an array of documents
df1 = df1.set_index("_id")
df1.head()
```

Out[4]:

	AvgAge	Count
_id		
China	46.250000	4
Antarctica	46.333333	3
Guernsey	48.333333	3
Puerto Rico	26.500000	2
Heard Island and Mcdonald Islands	29.000000	2

For simple cases one can either use a cursor through find("search term") or use the "$match" operator in the aggregation framework, like this:

In [5]:

```
pipeline = [
                {"$match": {"Country":"China"}},
]
aggResult = collection.aggregate(pipeline)
df2 = pd.DataFrame(list(aggResult))
df2.head()
```

Out[5]:

	Age	Country	Location	Name	_id
0	32	China	39.9127, 116.3833	Holman, Hasad O.	58d690f11ac4479b459dfdb3
1	43	China	31.2, 121.5	Byrd, Dante A.	58d690f11ac4479b459dfdee
2	57	China	45.75, 126.6333	Carney, Tamekah I.	58d690f11ac4479b459dfdf9
3	53	China	40, 95	Mayer, Violet U.	58d690f11ac4479b459dfe06

Use the MongoDB data

Let's do something with the data from the last aggregation, put their location on a map. Click on the markers to find the personal details of the four persons located in China.

In [6]:

```
import folium
print 'Folium version', folium._version_
```

```
world_map = folium.Map(location=[35, 100],
                                          zoom_start=4)
for i in range(len(df2)):
        world_map.simple_marker(location=df2.Location[i].split(','), popup=df2.Name[i]+', age:'+str(df2.Age[i]))
```

```
world_map
```

```
Folium version 0.2.0

/opt/anaconda/lib/python2.7/site-packages/ipykernel/_main_.py:7: FutureWarning: simple_marker is deprecated.
Use add_children(Marker) instead
```

Out[6]:

**In case no map is shown, try the following command from a terminal window and retry:

pip install folium or sudo conda install --channel https://conda.binstar.org/IOOS folium

Important Reference (For quick tips regarding pandas with mongodb follow the link):
http://ec2-54-218-106-48.us-west-
2.compute.amazonaws.com/moschetti.org/rants/mongopandas.html