Problem set2: Data Wrangling with MongoDB

Carrier List:

Carriers.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Please note that the function 'make_request' is provided for your reference only.
# You will not be able to to actually use it from within the Udacity web UI
# All your changes should be in the 'extract_carrier' function
# Also note that the html file is a stripped down version of what is actually on the website.
# Your task in this exercise is to get a list of all airlines. Exclude all of the combination
# values, like "All U.S. Carriers" from the data that you return.
# You should return a list of codes for the carriers.

from bs4 import BeautifulSoup
html_page = "options.html"

def extract_carriers(page):
  data = []

  with open(page, "r") as html:
    # do something here to find the necessary values
    soup = BeautifulSoup(html.read())
    carriers = soup.find(id='CarrierList')
    for eachoptiontag in carriers.find_all('option'):
      if len(eachoptiontag['value']) < 3:
        data.append(eachoptiontag['value'])

  return data

def make_request(data):
  eventvalidation = data["eventvalidation"]
  viewstate = data["viewstate"]
  airport = data["airport"]
  carrier = data["carrier"]

  r = requests.post("http://www.transtats.bts.gov/Data_Elements.aspx?Data=2",
            data={'AirportList': airport,
                'CarrierList': carrier,
                'Submit': 'Submit',
                "__EVENTTARGET": "",
                "__EVENTARGUMENT": "",
                "__EVENTVALIDATION": eventvalidation,
                "__VIEWSTATE": viewstate
            })

  return r.text
```

```
def test():
    data = extract_carriers(html_page)
    assert len(data) == 16
    assert "FL" in data
    assert "NK" in data

test()
```

Carriers.py Output: Congratulations, your solution is correct!

Airport_list:
airports.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# All your changes should be in the 'extract_airports' function
# It should return a list of airport codes, excluding any combinations like "All"

from bs4 import BeautifulSoup
html_page = "options.html"

def extract_airports(page):
    data = []
    with open(page, "r") as html:
        # do something here to find the necessary values
        soup = BeautifulSoup(html.read())
        carriers = soup.find(id='AirportList')
        for eachoptiontag in carriers.find_all('option'):
            if len(eachoptiontag['value']) < 4 and eachoptiontag['value'] != 'All':
                data.append(eachoptiontag['value'])
        print data

    return data

def test():
    data = extract_airports(html_page)
    print 'Length fo data: ', len(data)
    assert len(data) == 15
    assert "ATL" in data
    assert "ABR" in data

test()
```

airports.py Output:

```
['ATL', 'BWI', 'BOS', 'CLT', 'MDW', 'ORD', 'DFW', 'DEN', 'DTW', 'FLL', 'IAH', 'LAS',
'LAX', 'ABR', 'ABI']

Length fo data:  15
```

Processing_ALL:

process.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Let's assume that you combined the code from the previous 2 exercises
# with code from the lesson on how to build requests, and downloaded all the data locally.
# The files are in a directory "data", named after the carrier and airport:
# "{}-{}.html".format(carrier, airport), for example "FL-ATL.html".
# The table with flight info has a table class="dataTDRight".
# There are couple of helper functions to deal with the data files.
# Please do not change them for grading purposes.
# All your changes should be in the 'process_file' function
# This is example of the datastructure you should return
# Each item in the list should be a dictionary containing all the relevant data
# Note - year, month, and the flight data should be integers
# You should skip the rows that contain the TOTAL data for a year
# data = [{"courier": "FL",
#        "airport": "ATL",
#        "year": 2012,
#        "month": 12,
#        "flights": {"domestic": 100,
#                    "international": 100}
#        },
#        {"courier": "..."}
# ]
from bs4 import BeautifulSoup
from zipfile import ZipFile
import os

datadir = "data"

def open_zip(datadir):
    with ZipFile('{0}.zip'.format(datadir), 'r') as myzip:
        myzip.extractall()

def process_all(datadir):
    files = os.listdir(datadir)
    return files

def process_file(f):
    # This is example of the datastructure you should return
    # Each item in the list should be a dictionary containing all the relevant data
    # Note - year, month, and the flight data should be integers
    # You should skip the rows that contain the TOTAL data for a year
    # data = [{"courier": "FL",
    #        "airport": "ATL",
    #        "year": 2012,
    #        "month": 12,
    #        "flights": {"domestic": 100,
    #                    "international": 100}
    #        },
```

```
    #       {"courier": "..."}
    # ]
    data = []
    #info = {}
    #info["courier"], info["airport"] = f[:6].split("-")

    with open("{}/{}".format(datadir, f), "r") as html:

        soup = BeautifulSoup(html)
        table_rows = soup.find_all('tr', 'dataTDRight')
        for row in table_rows:
            table_cells = row.children
            cell_values = []
            for cell in table_cells:
                cell_values.append(cell.string)

            if "TOTAL" not in cell_values:
                info = {}
                info["courier"], info["airport"] = f[:6].split("-")
                info['year'] = int(cell_values[1])
                info['month'] = int(cell_values[2])
                info['flights'] = {}
                info['flights']['domestic'] = int(str(cell_values[3]).replace(',',''))
                info['flights']['international'] = int(str(cell_values[4]).replace(',',''))

                data.append(info)

    return data

def test():
    print "Running a simple test..."
    open_zip(datadir)
    files = process_all(datadir)
    data = []
    for f in files:
        data += process_file(f)
    assert len(data) == 399
    for entry in data[:3]:
        assert type(entry["year"]) == int
        assert type(entry["month"]) == int
        assert type(entry["flights"]["domestic"]) == int
        assert len(entry["airport"]) == 3
        assert len(entry["courier"]) == 2
    assert data[-1]["airport"] == "ATL"
    assert data[-1]["flights"] == {'international': 108289, 'domestic': 701425}
    print "... success!"
if __name__ == "__main__":
    test()
```

process.py Output: Running a simple test...
... success!

Patent Database:

patent.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# This and the following exercise are using US Patent database.
# The patent.data file is a small excerpt of a much larger datafile
# that is available for download from US Patent website. They are pretty large ( >100 MB each).
# The data itself is in XML, however there is a problem with how it's formatted.
# Please run this script and observe the error. Then find the line that is causing the error.
# You can do that by just looking at the datafile in the web UI, or programmatically.
# For quiz purposes it does not matter, but as an exercise we suggest that you try to do it programmatically.
# The original file is ~600MB large, you might not be able to open it in a text editor.

import xml.etree.ElementTree as ET

PATENTS = 'patent.data'

def get_root(fname):

    try:
        tree = ET.parse(fname)
        return tree.getroot()
    except ET.ParseError as e:
        print e
        line_number = [int(s.replace(',','')) for s in str(e).split() if s.replace(',','').isdigit()]

        with open(fname) as f:
            lines = f.readlines()
            print lines[line_number[0] - 1]

get_root(PATENTS)
```

patent.py Output:

```
junk after document element: line 657, column 0

<?xml version="1.0" encoding="UTF-8"?>
```

Lession 2: problem set2 : 5-Result of parsing the datafile

1)please enter content of the that is causing the error:
Ans: <?xml version="1.0" encoding="UTF-8"?>

2) What do you think is the problem?
Ans: After line 657 is another xml data which is duplicate and junk data that no use

Processing_patents:

Split_data.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# So, the problem is that the gigantic file is actually not a valid XML, because
# it has several root elements, and XML declarations.
# It is, a matter of fact, a collection of a lot of concatenated XML documents.
# So, one solution would be to split the file into separate documents,
# so that you can process the resulting files as valid XML documents.

import xml.etree.ElementTree as ET
PATENTS = 'patent.data'

def get_root(fname):
    tree = ET.parse(fname)
    return tree.getroot()

def split_file(filename):
    # we want you to split the input file into separate files
    # each containing a single patent.
    # As a hint - each patent declaration starts with the same line that was causing the error
    # The new files should be saved with filename in the following format:
    # "{}-{}".format(filename, n) where n is a counter, starting from 0.
    filenumber = 0
    startline = '<?xml version="1.0" encoding="UTF-8"?>'
    firstline = True
    #open file
    filecontent = ''
    f = open(filename, "r")
    while True:
        line = f.readline()
        #were at end of file so exit wjile loop
        if not line:
            break

        #if we find a newfile line, write current contents to file and start again
        if startline == line.rstrip("\r\n") and firstline is False:
            with open("{}-{}".format(filename, filenumber), "w") as of:
                of.write(filecontent)
            filecontent = ''
            filenumber += 1

        filecontent += line
        #set firstline to false after first line is read
        firstline = False
    #write final file
```

```
    with open("{}-{}".format(filename, filenumber), "w") as of:
            of.write(filecontent)
    f.close()
    return
    pass

def test():
    split_file(PATENTS)
    for n in range(4):
        try:
            fname = "{}-{}".format(PATENTS, n)
            f = open(fname, "r")
            if not f.readline().startswith("<?xml"):
                print "You have not split the file {} in the correct boundary!".format(fname)
            f.close()
        except:
            print "Could not find file {}. Check if the filename is correct!".format(fname)

test()
```

Split_data.py Output: Congratulations, your solution is correct!