Web Scraping I

ISTA 350 Hw6, Due 4/9/2020 at 11:59 pm

Introduction. This homework is the first part of an introduction to the art of web scraping. It is a three-part assignment. In the first part, you will grab the data you need from the web, put it into an html parser, and save the result into a file. Along the way, you will meet the `os`, `requests`, `bs4`, and `zlib` (or `gzip`) modules. **The test will take a really long time to run and generate a huge number of warnings. Just wait.**

Instructions. Create a module named `hw6.py`. Below is the spec for three functions and `main`. Implement them and upload your module to the D2L dropbox.

Testing. Download `hw6_test.py` and the associated files necessary for testing and put them in the same folder as your `hw6.py` module. Run it from the command line to see your current correctness score. Each of the four functions in `hw6.py` is worth 25% of your correctness score. The test file we will use to grade your program will be different and may uncover failings in your work not evident upon testing with the provided file. Add any necessary tests to make sure your code works in all cases.

Documentation. Your modules must contain a header docstring containing your name, your section leader's name, the date, `ISTA 350 Hw6`, and a brief summary of the module. Each method/function must contain a docstring. Each docstring should include a description of the function's purpose, the name, type, and purpose of each parameter, and the type and meaning of the function's return value.

Grading. Your module will be graded on correctness, documentation, and coding style. Code should be clear and concise. You will only lose style points if your code is a real mess. Include inline comments to explain tricky lines and summarize sections of code (not necessary on this assignment).

Collaboration. Collaboration is allowed. You are responsible for your learning. Depending too much on others will hurt you on the tests. "Helping" others too much harms them in reality. Cite any sources/collaborators in your header docstring. Leaving this out is dishonest.

Resources.
https://docs.python.org/3.7/library/os.html
https://stackoverflow.com/questions/27803503/get-html-using-python-requests
https://requests.readthedocs.io/en/master/
https://realpython.com/python-requests/
https://pypi.python.org/pypi/beautifulsoup4/
http://www.crummy.com/software/BeautifulSoup/bs4/doc/
http://www.zlib.net/manual.html
https://docs.python.org/3.7/library/zlib.html

`hw6.py`:

`get_soup`: this function has three parameters. The first is a string representing a URL and has a default argument of `None`. The second is a string named `fname` representing a filename also with default argument of `None`. The third is a Boolean named `gzipped` with a default value of `False`. `True` is passed to this parameter if the html to be parsed is gzipped. If the filename is not `None`, open

the file, pass the resulting file pointer to the `BeautifulSoup` constructor, and return the resulting object. Otherwise, if the url is `None`, raise a `RuntimeError` with the message `'Either url or filename must be specified.'` If it is not `None`, send a get request to the server. If the response content is zipped (third parameter), unzip it. Pass the content to the `BeautifulSoup` constructor and return the resulting object.

`save_soup`: this function takes two arguments, a filename and a soup object. Save a textual representation of the soup object in the file.

`scrape_and_save`: this function scrapes the following addresses, soupifies the contents, and stores a textual representation of these objects in the files `'wrcc_pcpn.html'`, `'wrcc_mint.html'`, and `'wrcc_maxt.html'`, respectively:
http://www.wrcc.dri.edu/WRCCWrappers.py?sodxtrmts+028815+por+por+pcpn+none+msum+5+01+F
http://www.wrcc.dri.edu/WRCCWrappers.py?sodxtrmts+028815+por+por+mint+none+mave+5+01+F
http://www.wrcc.dri.edu/WRCCWrappers.py?sodxtrmts+028815+por+por+maxt+none+mave+5+01+F

`main`: check the current directory for any one of the files that `scrape_and_save` creates. If it is not there, print `'---- scraping and saving ----'` and scrape and save the addresses. You can find out if a file exists in the current working directory using functionality provided by either the `os` module or the `os.path` module. Google them to learn more.