Homework Assignment 5

Posted: 4/14/2010
Due: 4/27/2010

The assignment is due in class; please follow the instructions on the homework submission form.

You are expected to study the Python tutorial and/or the book "Dive into Python", which are linked to from the course homepage.

Please use Python 2.6.

Objectives

- Explore object-oriented code reuse and Python's "batteries included" philosophy.
- Learn to download and install Python libraries.
- Write simple web tools in Python, using a number of libraries.
- Write a simple web application in Python using the web.py framework.
- Make use of a database from Python (extra credit).

Related Files

Homework submission form:
http://www.cs.unc.edu/Courses/comp524-s10/hw/submission-form.pdf

Part 1

Download and install the *Feedparser* library [1], and skim its documentation to get an overview of its functionality. Follow the installation instructions provided in the included `README` file.

Using the *Feedparser* library, implement a simple RSS/Atom command line application named `rss.py`. Your program should process the command line options provided in `sys.argv`.

Your program should download and output—in a human-readable manner—the latest news items for each of the feed URLs provided on the command line. The output should include the title, author, date, link, and a summary for each news item.

Your program should have the following features. Use the `optparse` module to realize command line argument processing.

- If provided with the flag `-h`, then `rss.py` should display a short help message explaining the available options and exit (try `python -h` for inspiration).
- If provided with the flag `-s`, then `rss.py` should omit each item's summary.
- If provided with the flag `-n` <*some number x*>, then `rss.py` should display at most *x* news items.
- If provided with the flag `-o`, then `rss.py` should open all (or *x*, if `-n` is specified) retrieved stories in a browser (using the `webbrowser` module).

For example, the command

```
python rss.py -s -n 5 -o http://news.ycombinator.com/rss
```

should display the title, author, link, and date of the top 5 stories on Hacker News (but not it's summary), and then open the link corresponding to each of the top 5 stories in a new tab of the system's default web browser.

Your program should handle malformed and invalid URLs gracefully, i.e., without crashing or hanging (delays due to network timeouts are ok).

Part 2

Write a Python program named `linkcheck.py` that accepts one or more URLs on the command line. The program should do the following with each provided URL:

- Download the specified HTML document using the `urllib` module.
- Parse the retrieved HTML document using the `htmlparser` module and extract all hyperlinks (i.e., find and process all `<a>` tags with an `href` attribute).
- Check for each link whether it is a "dead link", i.e., check that the linked-to document can be retrieved and that the contacted server does not return 404 as the HTTP status code [2]. Your program should support both relative and absolute URLs, i.e., your program should support links of the form `<a href="../index.html">`.
- Output a list of all links in the HTML document. Indicate for each link whether it is valid.

For example, the command

```
python linkcheck.py http://www.cs.unc.edu/Courses/comp524-s10/
```

should display all links on the course homepage and whether they are valid.

Your program should handle malformed and invalid URLs gracefully, i.e., without crashing or hanging (delays due to network timeouts are ok).

Part 3

Download and install the web.py framework [3]. Skim the tutorial and the code samples on the web page to explore basic web.py applications. (You can safely ignore the template and database connectivity features, but they may be beneficial for the extra credit extensions.)

Write a simple Python web application that mimics the basic Twitter [4] functionality using the web.py framework. In particular, your web application should offer the following features on separate web pages:

- Your application should offer a form to post new messages. This form should be available via the relative URL "`/new/`".
- Your application should offer a timeline of the so-far posted messages. The timeline should be available as the front page (relative URL "`/`").

Your program should persist the timeline on disk using the `pickle` module, and reload and display the stored timeline after a server restart (if any is available).

The goal for this part is to produce a working web application that can save and restore some state. Do not worry too much about the design of the web page (but it should be readable).

Extra Credit 1

Provide an appealing, clean, and simple web design for your solution for Part 3.

Extra Credit 2

Extend your solution for Part 3 to use the `sqlite3` module instead of the `pickle` module to persist the web application's state.

Extra Credit 3

Extend your solution for EC 2 to support multiple user accounts and a sign-up form, and store and display for each message the corresponding author.

Extra Credit 4

Extend your solution for EC 3 to support per-user RSS feeds of newly posted messages using the PyRSS2Gen library [5]. Test the generated RSS feeds with your solution to Part 1.

Guidelines

Your solution must be executable with `python` on the class host `stetson.cs.unc.edu`.

You may discuss possible approaches to all parts with other students, but you **cannot share source code**.

Please comment your code, and provide a short `README` document that explains how to run your solution to Part 3.

Deliverables

The Python source code for Parts 1–3 (and optionally the files for the extra credit problems).

Grading

Your solution will be predominantly graded on correctness.

35 points — Part 1.	10 points — Extra Credit 1.
30 points — Part 2.	20 points — Extra Credit 2.
35 points — Part 3.	20 points — Extra Credit 3.
	20 points — Extra Credit 4.

References

[1] Feedparser library: http://feedparser.org/.

[2] HTTP status codes: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

[3] web.py framework: http://webpy.org/.

[4] Twitter: http://www.twitter.com.

[5] PyRSS2Gen library: http://www.dalkescientific.com/Python/PyRSS2Gen.html.