ECE 364 Lab 12 Handout

November 27

Name: _____ ID: ee364_____ Email: _____@purdue.edu

Instructions

- You will work in the directory called `Lab12`.
- To submit, commit your files to SVN. ⚠ We will only grade the version of the file that is in SVN.
- ⚠ You must meet all base requirements in the syllabus to receive any credit.
- Use Python 3.4 (⚠ not 2.x, 3.3, 3.5, etc.) In PyCharm: File → Settings → Project Interpreter → Make sure that Python 3.4 (`/usr/local/bin/python3.4`) is selected.

⚠ Failure to meet all base requirements will result in zero credit. This includes submitting code that works with other versions of Python (e.g., 2.x, ≤3.3, ≥3.5, etc.), but not Python 3.4.

Rules: You may discuss the tools and metrics. You may use the web.

v2

Scrape Speaker Photos

This will get you started on Milestone 2 of the project. This is not the same as the project, but once you have done this, you should find some parts of Milestone 2 easier.

You are welcome to reuse code, but make sure to adjust your variable and function names.

Overview

You will write a function to scrape the web page of the Purdue Veterinary Medicine Speakers Bureau.

https://vet.purdue.edu/engagement/speakers-bureau.php

The function will fetch the photo of each speaker and store it in a given directory using the SHA1 checksum of the speaker's email address with an extension determined by the HTTP response header.

Requirements

Create a file called `scrape_speaker_photos.py` with a new function of the following form. This function is the sole product for Lab 12.

```
def scrape_speaker_photos(url, dir_path)
```

⚠ Do not hard-code the URL above. Your function should work on any web page with that structure, even if the URL, specific names of speakers, and/or email addresses are different. You may assume the general structure is the same (e.g., class attributes, the way elements are nested, etc.).

Here is the process your function will use to obtain and save the profile image files.

1. Fetch the HTML from `url`. (⚠ Reminder: Do not hard-code the URL on page 1.)

2. Parse the HTML using `lxml`.

3. For each speaker (person), find the `<img>` for their profile photo and their email address.

   You will need to use more `lxml` and (probably) XPath than you have needed so far. You should have received a reference sheet. There are many perfectly reasonable, correct ways to do this.

   Q: This sounds hard. Where do I start?
   A: Get to know the Developer Tools in your browser. Use right-click » *Inspect Element* to explore the structure of the document and attributes of each element.

   Q: Where can I use XPath expressions?
   A: Parameters called `path` on the lxml element reference take an XPath expression.

   Q: What is XPath?
   A: XPath is a language for selecting elements within the structure of XML and HTML documents.

   Q: Is XPath part of lxml?
   A: No. It is supported by many tools, including lxml.

   Q: The HTML is hard to read. Is there an easier way?
   A: In your browser, open *Developer Tools* and then choose the *Elements* tab (or similar).

   Q: XPath looks scary. How can I get started?
   A: Once you have finished step 2 (above), take a few minutes to experiment. Start with something like `etree.find(".//img")` or `etree.findall(".//img")`. Those fine all images. Then try something that looks at two levels of hierarchy at a time, such as `etree.find(".//div/img")` (any `<img>` inside a `<div>`). Also, try a query that looks for attributes, such as `find("div[@class='▒▒▒']")` (`<div … class="▒▒▒" …>`). *Some* of you may find this last form useful, but it depends on your overall approach.

   Q: Do I have to find everything with just one XPath expression (i.e., call to `etree.findall(…)`)?
   A: No. Once you have any node, you can query with that, too (e.g., `div_node.findall("…")`).

   Q: How do I find the email address, if I have the `<img>` node?
   A: In your browser, right click a profile image and *Inspect Element*. Then, find a path through the document tree structure to the email address. If you can't find it, use Inspect Element on the email address. Alternatively, instead of finding the `<img>`, you could see if there is some containing structure (e.g., `<div>`'s with specific attributes) that is consistent among all speaker profiles.

   Q: How do I distinguish `<img>` elements for profile photos from other images in the page?
   A: Use the structure of the page and/or specific element attributes. Use *Inspect Element* to examine profile images and then other images in Developer Tools. You will find that profile images are contained in a distinct structure of parent (and ancestor) elements that is different from any other images on the page.

   ⚠ *This is different from the project.* For this lab, you can assume that the general structure and attributes will be the same (even though URL, speakers, names, and email addresses may change).

4. Your function should fetch those images using `urllib.request` and save in the specified directory. The filename should be constructed as follows:

   SHA1 checksum of email address`.`*extension*

   Calculate *SHA1 checksum of email address* as described in the project description.

   Find *extension* using the `mimetypes.guess_extension(type)` function. Get `type` from the Content-Type header in the response object that was returned by `urllib.request.urlopen(…)`. In general, to get a specific header, use `resonse.info().get("▒▒▒▒")`.

   Q: How do I get the `src="…"` attribute from an `<img>` element?
   A: `node.get("src")`

   Q: `guess_extension(…)` returns weird extensions, like .jpeg and .jpe for JPG files. Is that okay?
   A: Yes. Use whatever extension it returns.

   Q: How do I save an image file?
   A: `urlopen(…).read(…)` returns a byte string. You just need to save that to a binary file. See the code quality standards for an example. (You can find it on Piazza.)

   Q: How do I combine dir_path with the filename?
   A: `os.path.join(dir_path, filename)`

Submission

Submit only your `scrape_speaker_photos.py` file (in Lab12). Do not submit any image files.