Lab Assignment 6 (CS229 only): Web Scraping Wikipedia

In this assignment you will practice web-scraping, by extracting information about programming languages from their wikipedia page. In particular we will be interested in three questions:

- What paradigms the language follows
- What type disciplines a language implements
- What languages influenced it
- What languages it influenced

You will be working with this script[1]. Download it and save it in whatever location you want to use.

Your assignment will be roughly split in 3 parts:

1. Read the list of programming languages with Wikipedia pages
2. Read individual information for each programming language
3. Ask various questions about the collection of programming languages

Collecting the list of programming languages

The `listPage` variable in the file, defined around line 16, contains the content of the Wikipedia page that lists all programming languages. You will need to process this list.

Start by viewing the contents of the page. You should do this in two ways. From a Python shell, you can do `print(listPage.prettify())`. And from a browser, you can open the page with the languages list[2], then open the developer tools of your browser, and in particular the "element inspector". Your goal is to find a way to identify the elements that form the various languages.

The script is not ready to be run. You will need to start a Python interpreter, and copy-paste parts of the script as you make progress. You should start by copy-pasting the first 19 lines of the script into your Python shell.

Look closely at the web-page contents. You should find that the lists that we are after are broken down by letter of the alphabet. Find the elements that define those sections. You should find them to be `div`s with some `class` attributes that you can target.

- **Exercise 1**. Line 23 has a start for computing a list of these sections, one for each letter of the alphabet. You will need to fix this line, by passing appropriate arguments to the `find_all` call, to target the required elements. If you have done this correctly, `len(letterSections)` should return 26.

[1] https://github.com/skiadas/DataWranglingCourse/blob/gh-pages/assignments/assignment6_229.py

[2] https://en.wikipedia.org/wiki/List_of_programming_languages

- **Exercise 2**. A skeleton for a function called getLanguagesFromSection is provided. It takes as input a section as those in letterSections and should return a list of the programming languages contains. It does this by a list comprehension, which produces a dictionary for every language. Your task is to fill in the values for this dictionary: It should have a "link" field, that contains the (relative) url for the language's page. You should be able to find this within the anchor link <a> within the list item that you iterate. It should also have a "name" field, that contains the name of the language. You should find that as the text contents of the <a> tag that is in the link. You should only need to add 2 lines.
- **Exercise 3**. We have provided the start of a list comprehension that computes the list of all languages. It should go over all the letterSections, and for each section go through all the languages returned by getLanguagesFromSection for that section, and make one long list of them all. You should only need to add 3 lines.

This concludes the first set of problems. At the end of the day you have a list of all languages for which Wikipedia has a page. In the next section we will reach into each of these pages and read out some of its information.

Read individual information for a language

We will now look into reading information for a particular language. As an example we will use the page for the C programming language. Make sure to open that page so you can see it.

- **Exercise 4**. In this exercise we have provided you with the skeleton for a function getSidebar that given a language object will return the sidebar on the corresponding page. There are two lines you need to fix. The first is the line computing the "langContents". This must use the getPage function that is defined near the top of the file, to access the link stored inside the language object. Take a look at the clang variable, defined in line 57, for how the lang argument to your function is supposed to look like.

  Next you will need to fix the call to "find" in the return line. That "find" needs to locate the sidebar. You should be able to identify the sidebar as being a table tag with a specific class argument. Look at the page in your browser, using the element inspector, to get an idea.

- **Exercise 5**. This does not require you to add any code, only to read code and comment on it. You are asked to read three functions and add comments, both above each function and if needed inline, to explain what these functions do. You may want to start with the larger function, as it calls the others. For that one, you can use the list produced by cSidebar.find_all("th") if you want to understand what the elements in the for-loop look like, and you can manually work and test with this list of elements, along with looking at the tag structure in the web-browser, to understand what is happening. Make sure you understand why there are double-uses of next_sibling at times.

- **Exercise 6**. In order to be able to do further work with the list of languages, we will turn it into a dictionary, called allLangDict. You will need to go through the allLanguages list, and populate the allLangDict dictionary as you go along. Each language should be placed in the object with key the link used for it, e.g. the C language's key should be '/wiki/C_(programming_language)'. A simple 2-line for-loop should suffice. This would allow us to follow for example a chain of influences from one language to another.

Work with the language structure

We will now perform various tasks on this list of languages.

We will start by looking at completeness of the master-list of programming languages, as follows: For each language in our list, we will look at its "influenced" list and its "influencedBy" list, and see if all the entries there are languages in our dictionary. If they are not, that means that Wikipedia does have an entry for the language but it does not have it listed in the "list of programming languages" page.

- **Exercise 7**. Loop over the allLanguages list. For each language loop over the languages it influenced and the languages that it was influenced by. If any of these languages does not appear as a key in the allLangDict dictionary, add it to the dictionary called missingLanguages and print the language link. Make sure to not print any language twice.

  Optional: Find out how Wikipedia pages can be updated, and help add the missing languages to the list.

We will now look for consistency between the influenced/influenced-by lists. For each language we will look at the languages it influenced, and make sure that each such language mentioned the original language in its "influencedBy" list. We will print any inconsistencies. Remember that the "influencedBy" list holds links.

- **Exercise 8**. Loop over the allLanguages list. For each language loop over the languages it influenced. Use allLangDict to look at each of these languages, and see if they mention the original language in their "influencedBy" list. Do a second loop the other way around. Remember to only look at languages that are not missing (i.e. that are in the allLangDict entries).

We will now turn our attention to the programming paradigms of the various languages, and the programming disciplines. You should study the results of these last two parts.

- **Exercise 9**. The dictionary allParadigms is supposed to contain as keys each programming paradigm and as value for a key it has an array of all the languages that follow that paradigm. Using a double loop over the languages and the paradigms in them, populate this dictionary.
- **Exercise 10**. Repeat with type disciplines. A typeDisciplines dictionary is initialized for you. I urge you to learn a bit more about some of these type disciplines.

Optional Questions

These are some optional extra questions. You should NOT submit these. But they are here in case you want to expand on this previous work. Instructions for submitting your homework are after this list.

1. (easy) missingLanguages contains a number of languages that we have not loaded into allLanguages. Do so. These new languages might then show you more new languages we had missed before. Add those and repeat until no new languages are produced.
2. (easy) Some language links appear with different capitalization when in the "influenced" or "influencedBy" arrays, as opposed to the "link" fields. Uniformize all cases by turning all three to lowercase.
3. (medium) Enhance the "influencedBy" and "influenced" fields in the languages objects based on the languages missing, as you found in exercise 8.
4. (easy) Order the languages based on how "influential" they've been, i.e. how many languages they have influenced. Similarly, order them by how many other languages influenced them.
5. (medium) For all pairs of distinct paradigms, compute the number of languages that have both paradigms, then sort the pairs based on that number, starting with the highest. This will show you which paradigms are most often combined.
6. (medium) Do the same for pairs of type disciplines.
7. (medium) This and the remaining problems require some knowledge of graph theory. We can consider the set of languages as a directed graph, where the vertices are the languages and an edge from language A to B means that language A influenced language B. We can represent this graph in two ways: As a list of (ordered) pairs (A, B), or as an object whose keys are the languages and the value for the key corresponding to language A is the list of all languages B that are influenced by B. The latter form already exists within the allLangDict dictionary. Use it to construct the former form.
8. (medium) Write a function that given a language A constructs the list of all languages that are (possibly indirectly) influenced by it. In terms of our graph picture this is the set of all edges that can be reached from A. You can use a recursive function, but beware of cycles.
9. (medium) Compute for each language/vertex the number of vertices that can be reached (as described in the previous problem). Then order the languages according to which reaches the most languages/vertices. You may either run the function you created in the previous part on all vertices, or you can try an algorithm that tries to compute them all at once as follows:

    - Start with a dictionary of numbers, with the languages as keys, initially populated by the value 0. The entries on the array are meant to determine how many languages each language influences.
    - For each entry, look at the languages it influences, add their numbers, and store it in the language entry. If it differs from the previously stored entry, remember that fact.

- Repeat the process until the numbers stabilize, i.e. until the last pass you make does not update any entries. If an entry exceeds the total number of languages, then that means your graph has a cycle.

10. (hard) Write a function that determines for any two languages/vertices whether they have a "common ancestor". This should include the case where one of the languages is an ancestor of the other.
11. (hard) Study a "topological sorting" algorithm and implement it on this graph. Topological sorting arranges the vertices in a linear order so that no edges "go backwards". Most standard topological sorting algorithms use depth-first-search techniques.
12. (hard) Compute the "strongly connected components" of this graph. The topological sort algorithm and some google search can help with that.
13. (hard) Find the longest path in the graph.
14. (hard) Implement some form of the PageRank[3] algorithm on this graph, and order the languages by their rank starting from the highest. For the purposes of this problem, use the "reverse" graph: If A influences B, then you should think of that as an edge from B to A. A language should gain value from the languages it influenced.

Submit

When you are ready to submit, make sure the assignment file you have been working on is saved and contains all the answers clearly marked, then email it to me.

[3] https://en.wikipedia.org/wiki/PageRank