CIT 590: Fall 2018
Homework 5

HW deadline as per Canvas.

This homework deals with the following topics:

- Reading and writing files
- Scraping information from a text file
- Very basic HTML

General Problem Specification

The basic skeleton of a website is an HTML page. This HTML page is a text file with a certain format. Taking advantage of this fact, one can take an HTML template and create multiple pages with different values stored. This is exactly what we are going to do in this HW.

Many websites use these kinds of scripts to mass generate HTML pages from databases. We will use a sample text file as our 'database'.

Our goal will be to take a resume in a simplistic text file and convert it into an HTML file that can be displayed by a web browser.

What is HTML?

You do not need to know much HTML to do this assignment. You can do the assignment by just understanding that HTML is the language that your browser interprets to display the page. It is a tag-based language where each set of tags provides some basic information for how the browser renders the text that you write between them.

For example, `<h1>The Beatles</h1>` will be rendered by your browser as a heading with large font. `<h1>` indicates the beginning of the heading and `</h1>` indicates the ending of the heading. An HTML webpage is typically divided into a head section and a body section. We have provided you with a basic website template. We want you to retain the head section and write only the body section via your Python program.

For more details about HTML, your best resource will be to use the w3schools website which can be found here: www.w3schools.com. This website has a ton of information and provides all of the common HTML you'll need to know for this assignment.

Input Test File

We will read a simple text file that is supposed to represent a student's resume. The resume has some key points but can be somewhat unstructured. In particular, the order of some of the information will definitely be different for different people.

The following are the things that you definitely DO know about the resume:

- Every resume will have a name, which will be written at the top. The top line in the text file will contain just the name.
- There will be a line in the file, which contains an email address. It will be a single line with just the email address and nothing else.
- Every resume will have a list of projects. Projects are listed line by line below a heading called "Projects". An example of what you might see in a resume file is something like this:

```
Projects
Worked on big data to turn it into bigger data
Applied deep learning to learn how to boil water
Styled web pages with blink tags.
Washed cars …
----------
```

- The list of projects ends with a single line that looks like '----------'. That is, it will have at least 10 minus signs. While this is a weird requirement, we are imposing it to actually make the assignment easier for you.
- Every resume will have a list of Courses. Courses are listed like:

```
Courses - CIT590, AB120
```

OR

```
Courses :- Pottery, Making money by lying
```

- You are allowed to assume that every single resume will just have this comma separated list of courses with the word "Courses" being there in front of them. You do want to allow for some kind of punctuation marks after the word "Courses".
- So your program should be able to look at the above example and then extract the courses without including the '–' sign or the '**:**–' or any such punctuation that is in between the word "Courses" and the actual data we're interested in. Every course begins with a letter of the English alphabet regardless of whether we are putting the course ID or the course name in our resume.

The first step in this assignment is to create a sample resume that conforms to the format described above. We're providing an example in the HW folder on Canvas, but please do not just blindly copy our resume. Write your own sample file in addition to the one provided. It does not have to be totally accurate but the HW is likely to be more fun if you make it close to reality.

Functions for Parsing the File

You need to write one function for each piece of information that you want to extract from this text file. Contrary to previous homework assignments, in this assignment we will not tell you what to call the functions or what arguments to pass to them.

What we will tell you is that you will be graded on how you name the functions, how modular your code is, and most importantly, whether you have unit tested your functions well or not.

Since the resume file is pretty small, these functions are best done by reading the file into memory and then parsing the file using list and string manipulations.

Reading the File

- Since the resume file is pretty small, write a function that reads the file and stores it in the program's memory.
- Then, you can use list and string manipulations to do all of the other necessary work.

Detecting Name

- This one is easy. Just extract the first line.
- The one extra thing we want you to do, just for practice, is to raise an error if the first character in the name string is not 'A' through 'Z'.
- So yes, in this case, your program will crash. You must provide a message saying that the first line has to be a name with proper capitalization.

Detecting Email

- Look for a line that has the '@' character.
- Also make sure that the last four characters of the email are either '.com' or '.edu'.

- Make sure the email string begins with a normal lowercase English character between the '@' and the ending ('.com' or the '.edu')
- There should be no digits or numbers in the email address.
- These rules will accommodate lbrandon@wharton.upenn.edu but will not accommodate @lbrandon or lbrandon@python.org or lbrandon2@wharton.upenn.edu
- We are fully aware that these rules are inadequate. However, we want you to use these rules and only these rules.
- PLEASE DO NOT GOOGLE FOR A FUNCTION FOR THIS. Googling for solutions to your homework is an act of academic dishonesty and in this particular case, you will get solutions involving crazy regular expressions, which is a topic we haven't yet discussed in class. In general, your code should never involve a topic that we have not discussed in class. Remember, if you already have expert knowledge of Python, you should not be in this class anyway.

Detecting Courses

- Look for the word "Courses" in the file and then extract the line that contains that word.
- Then make sure you extract the correct courses. In particular, any random punctuation after the word "Courses" and before the first actual course needs to be ignored.
- You are allowed to assume that every course begins with a letter of the English alphabet.

Detecting Projects

- Look for the word "Projects" in the file.
- Each subsequent line is a project, until you hit a line that looks like '----------'. This is NOT an underscore. It is (at least) ten minus signs put together. You have reached the end of the projects section if and only if you see a line that has at least 10 minus signs, one after the other.
- If you detect a blank line between project descriptions, ignore that line.

Writing HTML

Once you have gathered all the pieces of information from the text file, we actually want you to programmatically write HTML. Here are the steps for that:

- Save the file *resume.html* that is provided in the same directory as your code.
- Open that file in a text editor that does HTML syntax highlighting (e.g. Sublime Text). Notice that there is empty `<body>`. We are going to programmatically open this file and fill the body.
- In order to do this, your Python code will open the HTML file that we have provided and then write information to it bit by bit. In order to do that, you will probably need the following code. Note that you will have to change the path so that it points to wherever you put your `resume.html` file.

```
f = open('path/to/file.html', 'r+')
lines = f.readlines()
f.seek(0)
f.truncate()
del lines[-1]
del lines[-1]
f.writelines(lines)
```

- `f.seek(0)` and `f.truncate()` delete everything from the file after we store the file contents in lines. Then we are deleting the last two lines by running `del lines[-1]` twice and writing everything back to the file (except for the last two lines) using `writelines`.

- Why are we doing this? Because our file looks something like this and we need to start by removing the last two lines (body and html tags).

```
random header stuff
<html>
<head> lots of style rules we won't worry about
</head>
<body>
</body>
</html>
```

- We want to put our resume content in the body tags to make it look like this:

```
random header stuff
<html>
<head> lots of style rules we won't worry about
</head>
<body>
all the resume content goes here
</body>
</html>
```

- In order to write proper HTML we will need to write the following helper function:

  surround_block(tag, text):
  - This function surrounds the given text with the given HTML tag and returns the string
  - For example, *surround_block('h1', 'The Beatles')* would return *'<h1>The Beatles</h1>'*

Now break the writing of the resume into the following steps

1. In order to format the resume a little bit we have to make sure that all the text is enclosed within the following lines.

   ```
   <div id="page-wrap"> </div>
   ```

   Since we can only write things line by line we will write the first line now, then fill up the actual content in a few steps and finally write 3 lines to close the HTML file properly. So this initial step just writes one line which is:

   ```
   <div id="page-wrap">
   ```

2. The basic information section needs to look like

   ```
   <div>
   <h1> Brandon </h1>
   <p> Email: lbrandon@whraton.upenn.edu </p>
   </div>
   ```

   Think about how you can do that. In particular think about how the *surround_block* function can be used to achieve this. Write a function to make this intro section of the resume and then write it out to a file.

3. For the projects section you will have to produce output like the following. Assume that you have got the data about the projects by reading through the original file and stored that in some data structure (decide whether you want to use list, set, tuple or dictionary)

   ```
   <div>
   <h2>Projects</h2>
   <ul>
   <li>built a robot</li>
   <li> fixed an ios app </li>
   </ul>
   </div>
   ```

4. For the courses section, we actually just want to list the courses. We do not want to create bullet points. We also want the heading to be a bit smaller in size. So here is an example of generated HTML:

```
<div>
<h3> Courses </h3>
<span> Algorithms, Race car training </span>
</div>
```

   Write a function that takes in courses (you can decide whether you want a list of courses or a string of courses) and then writes out the HTML, in the form above, to the file.

5. After writing all of this content we just need to remember to close the HTML tags. To do this, we need to write the following 3 lines to the file:

```
</div>
</body>
</html>
```

6. And most importantly, you need to close the file -- otherwise it will NOT save!

What to Submit

You will submit the following 4 files:

1. *make_website.py*: the program you wrote
2. *test_make_website.py*: the unit tests you wrote for all your functions
3. *resume.txt*: the text file you created to be read by your program
4. *resume.html*: the resume file that was generated when you ran your program

Please zip all 4 of these files and name the .zip file using your PennKey. For example, Brandon's submission would be *lbrandon.zip*. Please, no .rar files!

Evaluation

It is important that you understand that in this assignment there is more to it than just getting your code to work. We are happy to give you feedback about your design during office hours.

1. Functionality – 15 points

    a. Does your program successfully convert a resume text file into a resume HTML file?
    b. Can that HTML file be rendered in a web browser? Test it out in at least 2 different browsers to confirm this.
2. Unit Testing – 10 points
    a. Did you write a test for each function you wrote?
    b. Are you doing a good job of testing all the different cases for each function?
3. Design and Style – 15 points
    a. Since we do not give you specific functions, we need to evaluate your design.
    b. Did you follow style conventions as defined in this course?
    c. Did you use the best data structures and variable types for each situation?
    d. Did you simplify the logic of your functions? In other words, if it is possible to do the same thing with fewer lines of code, did you do so?