UPRM CSE Department

CIIC 3011-Introduction to Computer Programming I

Fall 2019-2020

Project 2: HTML Resources Parser

Submission date: October 23, 2019

Learning objectives

- Create computational artifacts.
- Apply programming control structures (sequence, selection, and repetition).
- Process text files.

Motivation

You're a newly-assigned webmaster for TechInfo, a company that provides technology services in the aerospace industry. Trying to cut costs on their web maintenance contract, TechInfo's main customer has adopted a content management system (CMS) and has copied all of their content to the new CMS. TechInfo now has the task to archive the old website, including all of the internal resources (e.g. other files like images and scripts) used by each webpage.

Your Task

You must create a program that parses an HTML file and finds all of the internal resources used by that file. The results must then be saved in an output file, which would then be used as a guide to know all of the files that need to be removed.

Technical Details

HTML 101

An HTML file consists of *tags* that instruct the browser how to display text. Most tags have an opening tag of the form of `<tagname>`, and a closing tag of the form `</tagname>`, where *tagname* is replaced by the name of a specific tag. For example, `<em>I want this emphasized</em>` would be displayed by the browser as *I want this emphasized*, and <strong>This should be strongly emphasized</strong> would be displayed as **This should be strongly emphasized**. There are a few tags that do not have a closing tag.

Most HTML tags use *attributes* to convey additional information to the browser. A hyperlink must not only inform the browser of its existence, but also to what page or resource is being linked to. For example, in the tag `<img src="logo.png">`, the attribute `src` tells the browser where to find the file to be used as an image.

Tags that specify resources

Although HTML provides the content structure, the appearance is designed through Cascading Style Sheets (CSS) and the behavior is typically programmed through JavaScript (JS). Additionally, most pages use images and also have hyperlinks to other webpages. All of these resources are referenced with the following tags (hereby referred to as the “tags of interest”):

- `<a>` Hyperlink to another page.
  Example: `<a href=”homepage.html”>Click here to go home!</a>`
- `<script>` Imports an external JavaScript file or defines JavaScript code
  Example (external file): `<script src=”funkyscript.js”></script>`
  Example (code): `<script>alert(“Hello world!”)</script>`
- `<link>` Imports an external CSS document (among other things).
  Example: `<link href=’awesomestyle.css’>`
- `<img>` Includes an image.
  Example: `<img src=”logo.png”>`

Note: There are other tags that specify resources, such as `audio` and `video`, but we will ignore those for now. Also note that `link` and `img` do not have a closing tag.

Focus on starting tags

Although some tags do not have a closing tag, that doesn’t matter, because the resource is always in the opening tag. In the tag `<a href=”home.html”>Click here to go home!</a>`, the actual resource that we’re interested in is *home.html*. In the tag `<img src=”logo.png”>`, the actual resource that we’re interested in is *logo.png*. Note that the resource in the `a` and `link` tags are inside the `href` attribute, whereas in the `script` and `img` tag it is inside the `src` attribute. Also note that it’s possible to find a tag without the attribute of interest (see the second `<script>` example above), in which case the tag is to be ignored.

Input / Output

- Your program should automatically read a file with the name **index.html** as input. The input file should be in the same directory as your program. *Your program must not prompt for the file name.* A sample index.html file is provided to be used for testing.
- Your output should be saved in a file with the name **index_resources.txt**, which must be created in the same directory as the input file.
- The resources must be grouped in one of the following four categories: CSS, JavaScript, Images, Hyperlinks (in that order).
- Every category should first list its title followed by a colon (‘:’) on a single line, followed by its resources *in alphabetical order*. If there are no resources for a particular category, the category name should not be printed.

Your Code:

- Use the provided code and maintain the structure.
- Use of lists and lists methods to save and process the tags and corresponding resources to achieve the assigned task.

Additional details

- An HTML file is simply a text file that has the .html extension. You don't need to learn html to do this project.
- For simplicity, assume that there's at most one *tag of interest* per line, although there may be whitespace at the beginning of the line since HTML is usually indented for readability.
- You may also assume that all tags and their attributes are in lowercase, and that attribute values are enclosed in either single or double quotes (no need to validate this).
- It is possible for the tags to have other attributes; *you must not assume that immediately after the tag name you will find the attribute of interest*.
- Don't look directly for the attributes, only look for the attributes within a tag of interest.
- Because our goal is to archive local files, we can ignore all external resources (i.e. those that start with http: or https:)
- You must only open the input file once for processing; don't close and re-open it.
- You cannot assume anything about the name of the resources, nor their location. You cannot assume that they will have the same names as in the examples, or that they will be located inside folders as in the example.

Example:

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>HTML Resources</title>
    <link href="css/html101.css" rel="stylesheet">
    <script src="js/html101.js"></script>
</head>
<body>
    <header>
        <img id="imgRUM" src="img/logo_rum.png" alt="Logo RUM">
        <h1>HTML Resources</h1>
    </header>
    <main>
        <p>
                Here's a list of the tags of interest:
                <ul>
                        <li>a</li>
```

```
                    <li>img</li>
                    <li>link</li>
                    <li>script</li>
              </ul>
        </p>
        <p>
              The attributes of interest, with the tags they correspond to:
              <ul>
                    <li>src: img, script</li>
                    <li>href: link, a</li>
              </ul>
        </p>
        <p>Use only attributes inside tags, not this one: a href=”here.html”</p>
    </main>
    <footer>
        <p>&copy; 2019 UPRM</p>
        <p><a href="https://cse.uprm.edu/" target="_blank">CSE Department</a></p>
    </footer>
    <script src="js/cse_functions.js"></script>
</body>
</html>
```

`index_resources.txt`

```
CSS:
css/html101.css
JavaScript:
js/cse_functions.js
js/html101.js
Images:
img/logo_rum.png
```

Note that the only hyperlink was to an external site, so we didn’t include it in our output. Since we found no internal hyperlinks, the output didn’t include the “Hyperlinks:” section. Also, there was more than one JS file, so the resources in that section are listed in alphabetical order.

Submission details

You must submit the .py file for your project through the appropriate assignment entry in Moodle under the Project 2 section. Your file name must have the following format: `LastnameFirstname_Section_P2.py`. For example, a student named Juan Lopez from section 016 would submit the following file: `LopezJuan_016_P2.py` (please use only one name and one last name).

Project due date: October 23, 2019

NOTE: Projects that don't execute correctly on the Python interpreter will get a score of 0.

Last modified: October 11, 2019 5:15PM