# A9: XHTML

* Assignment A9 should be complete individually or with one partner.
* If you work with a partner, be sure to follow good pair programming practices.
* Be sure to read the entire prompt and understand the problem before beginning coding.

## Learning Objectives

* Work with both input and output files
* Gain additional practice working with strings
* Learn a bit about web page structure

## Web pages and their Structure

Why are we talking about web page structure? You will write a Python program that will give you an idea how a program like a compiler can be written to check, identify, and even correct the syntax of incorrect code. Your program will use the source code of an HTML document as input and will modernize the uses of particular HTML tags, writing an improved version as output.

If you "view source" on any web page, you will see the code which is used to render the page in the browser. This code is created in a language called HTML (HyperText Mark-up Language) or possibly XHTML (eXtensible HyperText Markup Language) which is a family of XML markup languages that were developed to extend versions of the more widely supported HTML. HTML language standards have changed over time from HTML 1.0 to HTML 4.1 and the HTML 5.0 which was just finalized and released in October 2014. See <http://www.w3.org/TR/html5/> if you are interested in finding out more.

Click on [basic.html](http://cs.berea.edu/courses/2015/csc226/tasks/basic.html) and then view source. When you view source you will see:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta content="text/html;charset=UTF-8" http-equiv="Content-Type">

<title>Basic Web Page</title>

</head>

<body>

<h3>A Basic Web Page</h3>

<p>A basic web page looks something like this. It might have

images like:<br>

<img style="width: 66px; height: 44px;" alt="WC3 image" src="wc3.gif">

<br>

And it might have a line like:

</p>

<hr>

<p>Be sure to "view source" of this page.</p>

</body>

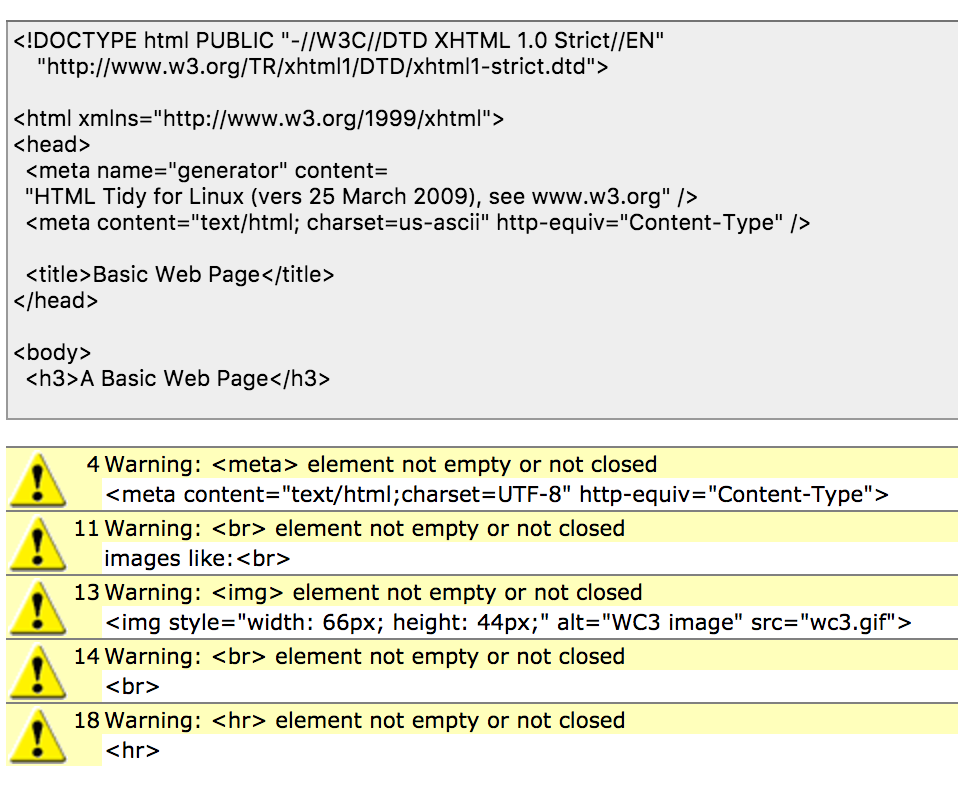
</html>

You can also save or copy this HTML in case you want to use it for testing your program. :)

There doesn't seem to be much wrong with this simple web page. It renders correctly in your browser, right? Except that it has gotten old and has some elements which need to be modernized.

Go to <http://infohound.net/tidy/> and paste in either the source code or the link of [http://cs.berea.edu/courses/csc226/tasks/basic.html](http://cs.berea.edu//courses/csc226/tasks/basic.html). Note that it can also take a file, which will be useful later in this assignment.

You will see that errors are identified, like below:



What is going on?

Most of the tags in HTML have opening and closing parts:

<html></html>

<head></head>

<title></title>

<body></body>

<h1></h1>, <h2></h2>, <h3></h3>, <h4></h4>, <h5></h5>, and <h6></h6>

etc.

Those tags do not have problems in this HTML code. The problems come with certain tags which were not required to close in early HTML because XHTML requires them to close, and that is why there are these errors.

Some of the most common HTML tags which used to not require closure are:

<meta ...>

<img src="filename" ...>

<br>

<hr>

So, we can fix the basic web page by fixing these as follows: we add a " " followed by a "/" right before the ">".

<meta ...> becomes <meta ... />

<img src="filename"...> becomes <img src="filename"... />

<br> becomes <br />

<hr> becomes <hr />

Click on [basic-fixed.html](http://cs.berea.edu/courses/2015/csc226/tasks/basic-fixed.html), which will look identical, until you view source which has these important changes, marked in red:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta content="text/html;charset=UTF-8" http-equiv="Content-Type" />

<title>Basic Web Page</title>

</head>

<body>

<h3>A Basic Web Page</h3>

<p>A basic web page looks something like this. It might have

images like:<br />

<img style="width: 66px; height: 44px;" alt="WC3 image" src="wc3.gif" />

<br />

And it might have a line like:

</p>

<hr />

<p>Be sure to "view source" of this page.</p>

</body>

</html>

Go to <http://infohound.net/tidy/> and paste in either the link of [http://cs.berea.edu/courses/csc226/tasks/basic-fixed.html](http://cs.berea.edu/courses/csc26/tasks/basic-fixed.html)

or the source itself.

You will see similar results to before, without any yellow errors at the bottom.

Yay! All fixed! But, what if there had been LOTS of these errors... then we would not want to have to change them by hand.

## Your task

In this assignment, it is recommended that you begin by designing your program with a partner. Then you may work with a partner or individually for the implementation. In this assignment, you will modernize HTML source code by writing Python code to update HTML tags which did not originally require closure.

Specifically, your Python program should:

1. Prompt the user to type in the name of the input html file to be corrected (e.g. basic.html).
   * If this input file does not exist, your program should quit gracefully (i.e. output a message saying what is wrong before quitting).   
       
     UPDATE: 03/24/17: Step 1 asks for user input, then validate the file exists. You don't know how to do that yet! Here's the solution to Step 1:

filename = input("Enter the filename: ")

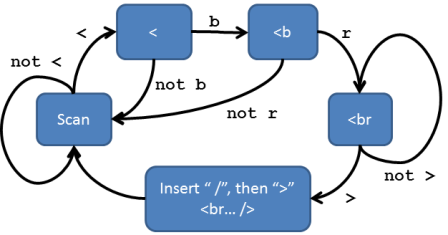
try:

file = open(filename, "r")

except:

print("Invalid file name")

* + Note that it is easiest to have this file in the same folder as the program or else you need to type the path to finding it.
  + See the examples from T11 and revisit the reading on Files for examples on how to read and write to files.

1. Prompt the user to enter the name of the file to write the corrected version (e.g. basic-fixed.html). You should assume that this filename will NOT be the same as the input filename.
2. Create one or more functions which do the HTML corrections by adding " /" (space+slash) right before the ">" in the following tags:
   * <meta ...> should become <meta ... />
   * <img src="filename"...> should become <img src="filename"... />
   * <br> should become <br />
   * <hr> should become <hr />
3. Some notes:
   * Note that it is possible for HTML to have many of each type of these old style HTML tags, and they can appear in no particular order. You should handle this in your code.
   * Note also that some of these tags have a large amount of "stuff" called attributes which you need to leave alone... you will just need to find the next ">".
   * Note that you may assume that the "<" followed by two matching characters uniquely identifies the tag. In other words, if you find <me it is <meta ...>; if you find <im it is <img ...>; if you find <br it is <br >; and if you find <hr it is <hr>.
   * You MAY also assume that none of these four specific tags have already been updated in old html, so you do not need to worry about checking to see if there is already a " /" before the ">" of these tags. The " />" should be added ONLY at the end of the four specified tags
   * How you use Python strings features can make this assignment easier or harder, so think about the design before you start coding! [Revisit the Python documentation on String methods for the exhaustive list of things you can do with strings.](https://docs.python.org/3/library/stdtypes.html#string-methods)
4. As you think about how to design your program, you might consider what state the program is in based on what characters it has encountered so far.
   * The default state could be "Scan" for characters, in which all it does is continually look at the next character.
   * Suppose you are looking for a <br>. The program transitions out of the "Scan" state when it encounters a <, in which case it knows that an HTML tag has started (any other character would tell it to keep going).
   * At this point, if it encounters a b, it is now in the state of being inside the <b tag; any other character would indicate "False Alarm" and revert it back to the "Scan" state.
   * If the program next encounters an r, then it enters the state indicating that it is inside a <br tag. Again, any other character than an r would indicate that we did not find a <br, so we go back to the "Scan" state.
   * When the program next encounters the >, it is almost done with the <br> tag. It needs to add a space followed by a / right before the >. Hence, the loop marked by "not >", allows the program to stay inside the <br tag, continually scanning the next character waiting for the > to appear. Once the space /> is added, then the program can move back to the original "Scan" state.
5. The image below demonstrate these transitions:
6. 
7. Of course, you can use this same kind of logic (using if to transition from one state to the next) to scan for any of the other tags for this assignment.
8. Have your program display the corrected HTML on the screen and also inform the user of how many of each type of correction your program made.
9. Your program must have good structure and style:
   * It must include a main() function.
   * The highest level of the program (i.e., no indenting) **must only** contain the following:
     1. the header
     2. any import statements
     3. function definitions
     4. a call to the main() function
   * You MUST use functions for encapsulation. Your program must be designed in a modular fashion, correctly using functions for each task with correct parameter passing and appropriate use of returns.
   * Use only meaningful variable and function names.
   * Insert a descriptive docstring for each function you are designing and implementing.
   * Use comments to clarify sections of code and include the header information, and include a descriptive header as a comment at the top of your source code.

## Submission Instructions

1. Review the requirements above to ensure you have completed everything that was required of you.
2. Save your code as **A9\_XHTML\_*username*.py**. Replace *username* with your Berea usernames. For example, the TA Bianca Marrero’s file would be **A9\_XHTML\_marrerob.py.**
3. Upload the file to Moodle by the due date listed on the course website: <https://trello.com/b/w7bIrLoV/>.
4. If you worked with a partner, your partner should upload a file named **A9\_XHTML\_*usernames*.txt** and include both partner’s name in the document.