Name:_____ **Partners:** _____

Python Activity 70: Web Scraping with Python

Python is a ***powerful tool*** that processes & manipulates, data; accesses data; creates beautiful things such as art, solutions, and puzzles; and expands human capabilities. But it also ***communicates complex computational ideas***!

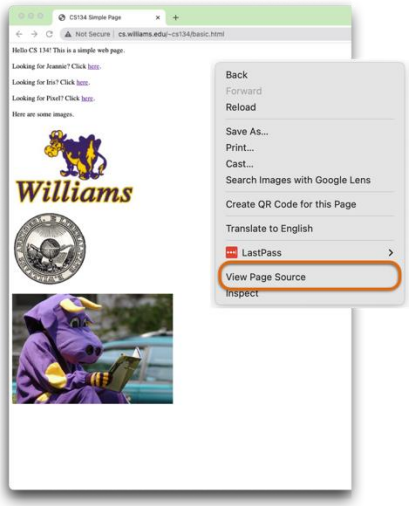
Learning Objectives
Students will be able to:
Content:
- Describe what **HTML** is
- Define some common **HTML tags**
- Predict what code using the beautifulsoup4 module will do

Process:
- Write some basic **HTML code**
- Write python code that scrapes web data

Prior Knowledge
- Python concepts: modules, methods

Critical Thinking Questions:

1. If we open a web page (cs.williams.edu/~cs134/basic.html) with a web browser, and then right-click the page and select "View Page Source" as below on the left, a new window opens with the text on the right (only first 11 lines shown):



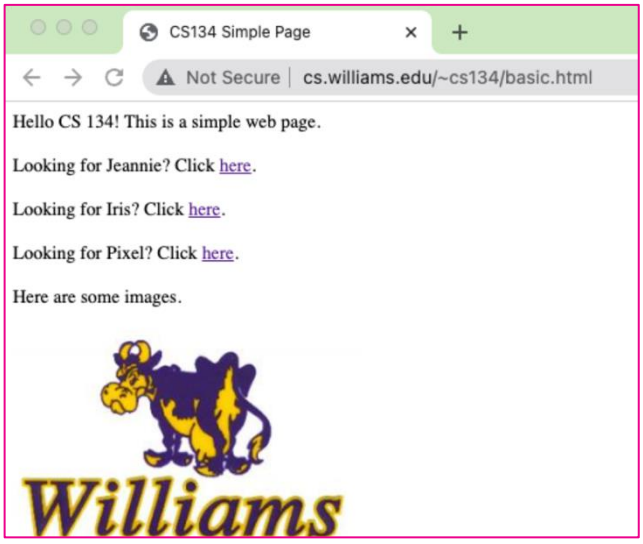
basic.html

```
<html>
  <head>
    <title>CS134 Simple Page</title>
  </head>
  <body>
    Hello CS 134!  This is a simple web page.

    <br><br>
    Looking for Jeannie?  Click <a
href="http://www.cs.williams.edu/~jeannie">here</a>

    <br><br>
    Looking for Iris?  Click <a
href="http://www.cs.williams.edu/~iris">here</a>.
```

a. If we change the line "`Hello CS 134! This is a simple web page.`" to "`<font color="blue">Hello CS 134! This is a simple web page.</font>`" what do you think will happen to the webpage (zoomed-in, below)?:



b. The text, "`Hello CS 134! This is a simple web page.`" will turn blue .What do you think the "`<font color="blue">`" tag does?

c. What do you think the "`</font>`" tag does?

d. How might we modify the "`<font color="blue">`" tag to make text *green*?

FYI: ***Hypertext Markup Language****, or* **HTML**, is a markup language (not a programming language!) that specifies how to format text for your Internet browser. Different tags/symbols specify how the computer should display the text.

2. Match the HTML tags on the left with what you think they might do on the right:

`<h1>Text goes here</h1>`	Makes the text bold (or: `<strong>`)
`<b>**Text goes here**</b>`	Changes the font type
`<i>*Text goes here*</i>`	Makes a hyperlink
`<a href="http://url-here.edu">Text here</a>`	Changes the text's color
`<font face="courier">Text here</font>`	Changes the text's size
`<font size="+2">Text goes here</font>`	Inserts ~2 newlines (paragraph definition)
`<font color="green">Text goes here</font>`	Inserts ~1 newline (line break)
`<p>Text goes here</p>`	Makes a level 1 heading
` `	Makes the text italic (or: `<em>`)

3. Well-formed HTML files also need a bit of structure. Match the structural HTML tags on the left with what you think they might do on the right:

```
<html>
<head>Text & Tags go here</head>
<title>Text goes here</title>
<body>Text & Tags go here</body>
</html>
```

Specifies text to appear at top of web browser

HTML in here is the main part of the file

Ends the HTML file

Defines what markup language is being used

HTML in here is the metadata for the file

4. We can import a new module, `requests`, and use its methods as below:

Interactive Python

```
>>> import requests
>>> r = requests.get('http://www.cs.williams.edu/~cs134/basic.html')
>>> r.text

'<html>\n  <head>\n    <title>CS134 Simple Page</title>\n  </head>\n\n  <body>\n
Hello CS 134!  This is a simple web page.\n\n    <br><br>\n    Looking for
Jeannie?  Click <a href="http://www.cs.williams.edu/~jeannie">here</a>.\n\n
<br><br>\n    Looking for Iris?  Click <a
href="http://www.cs.williams.edu/~iris">here</a>.\n\n    <br><br>\n    Looking for
Pixel?  Click <a
href="https://www.cs.williams.edu/~iris/website/img/HAILab.jpg">here</a>.\n\n
<br><br>\n    Here are some images. \n\n    <br><br>\n    <img
src="http://sysnet.cs.williams.edu/Williams-Logo.jpg" alt="purple cow">\n\n
<br><br>\n    <img src="http://sysnet.cs.williams.edu/williams.gif" alt="seal">
```

a. What is 'http://www.cs.williams.edu/~cs134/basic.html'?
Hint: How does it relate to the webpage in the previous questions?

How does the text displayed by `r.text` relate to the HTML file we saw in question1?

b. What do you think the `requests.get(..)` method does?

What do you think the `r.text` attribute contains?

Is what's stored in `r.text` easily readable? _____

5. We can import a new module, `beautifulsoup`, and use its methods as below:

Interactive Python (continued)

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(r.text, 'html.parser')
>>> print(soup.prettify())

<html>
 <head>
  <title>
   CS134 Simple Page
  </title>
 </head>
 <body>
  Hello CS 134!  This is a simple web page.
  <br/>
  <br/>
  Looking for Jeannie?  Click
  <a href="http://www.cs.williams.edu/~jeannie">
   here
  </a>
  .
  <br/>
  <br/>
  Looking for Iris?  Click
  <a href="http://www.cs.williams.edu/~iris">
```

a. How does the text displayed by `print(soup.prettify())` relate to the `r.text` command we saw previously?

_____

Is what's returned by `print(soup.prettify())` easily readable? _____

b. What do you think the `BeautifulSoup` module does?

_____

FYI: Oftentimes, data we obtain is not in a meaningful format. To convert raw text data into more meaningful data, we must ***parse*** the data. We could, of course, build our own custom string manipulation methods to implement this behavior, but there's plenty of existing Python modules that will do this for us, of which ***BeautifulSoup*** is one.

6. Observe these additional interactions in interactive Python that continue from our previous example:

Interactive Python

```
>>> soup.title
<title>CS134 Simple Page</title>
>>> soup.title.name
'title'
>>> soup.title.string
'CS134 Simple Page'
>>> soup.title.parent.name
'head'

>>> soup.img
<img alt="purple cow" src="http://sysnet.cs.williams.edu/Williams-Logo.jpg"/>

>>> soup.a
<a href="http://www.cs.williams.edu/~jeannie">here</a>
>>> soup.find_all('a')
[<a href="http://www.cs.williams.edu/~jeannie">here</a>,
 <a href="http://www.cs.williams.edu/~iris">here</a>,
 <a href="https://www.cs.williams.edu/~iris/website/img/HAILab.jpg">here</a>]
```

a. What *type* of object might `soup` be?

_____

What does the `soup.title` attribute contain?

_____

What *type* of object might `soup.title` be?

_____

b. What does the `.name` attribute contain? *Hint: it does this for all HTML tags!*

_____

What does the `.string` attribute contain? *Hint: it does this for all HTML tags!*

_____

c. What does the `soup.img` attribute contain?

_____

What might `img` be an abbreviation for? _____

d. What does the `soup.a` attribute contain?

_____

What does the `soup.find_all('a')` method return?

_____

7. Observe these additional interactions in interactive Python that continue from our previous example:

Interactive Python

```
>>> for link in soup.find_all('a'):
...         print(link.get("href"))

http://www.cs.williams.edu/~jeannie
http://www.cs.williams.edu/~iris
https://www.cs.williams.edu/~iris/website/img/HAILab.jpg

>>> for link in soup.find_all('a'):
...         print(link.get_text())

here
here
here
```

a. What does the above code do?

_____

_____

_____

_____

8. Name at least three reasons why we might want to pull source code from web pages:

_____

_____

_____

Application Questions: Use the Python Interpreter to check your work.

1. Explore the BeautifulSoup documentation: https://beautiful-soup-4.readthedocs.io/en/latest/
   What other methods are there?

2. Write a function, `maybe_names(url, valid_words)`, that returns all invalid words found in a given webpage located at `url`. Assume that `valid_words` is a list of strings containing all valid words.