Finishing up with Python.

1. No hours this week; no class on Friday; HW due Monday.

2. Questions?


   (a) Install with

   ```
   pip install beautifulsoup4
   ```

   (b) Good documentation at www.crummy.com/software/BeautifulSoup/bs4/doc.

   (c) Typical usage:

   ```python
   >>> import requests
   >>> from bs4 import BeautifulSoup
   >>> html = requests.get('http://www.cs.williams.edu/~bailey/cs135').text
   >>> soup = BeautifulSoup(html, 'html.parser')
   >>> soup.prettify()
   ```

   Here, `soup` is the entire document tree. For a typical page, there are many subtrees.

   (d) Given a soup/tree you can print it nicely with

   ```python
   >>> print(soup.prettify())
   <html>
   <head>
   <link href=''index.css'' rel=''stylesheet'' type=''text/css'">
   <title>
   Computer Science 135 - Diving into Python
   </title>
   </link>
   </head>
   ```

   (e) Given a tree, you can get subtrees identified by `tag`, `tag`, with `soup.tag`. For example:

   ```python
   >>> print(soup.title.prettify())
   ...
   ```

   It finds just the first instance of that tag.

   (f) To find a subtree by id, use `t.find(id='homeworks')`:

   ```python
   >>> soup.find(id='homeworks')
   '<p class="title" id="homeworks">...
   ```
(g) To get a list of subtrees use, for example, `t.find_all('a')`. Typically, you iterate across the list.

(h) To get the tag associated with a tree, use `t.name`. It always returns a string, the tag.

(i) To get the text contents of a text-based tag, `t`, use `t.string`.

```python
>>> soup.a.string
'Lectures'
```

(j) To get the text of a particular key associated with a tag, use dictionary indexing:

```python
>>> soup.a['href']
'#lectures'
```

(k) To get the printable text associated with a soup/tree, use `get_text()`.

4. Quines: programs that print themselves...

    *

    *