Decode A Web Page 👣 👣 👣 👣

Exercise

Use the `BeautifulSoup` and `requests` Python packages to print out a list of all the article titles on the New York Times homepage.

Discussion

Concepts for this week:

- Libraries
- `requests`
- `BeautifulSoup`

Libraries

Many people have written libraries in Python that do not come with the standard distribution of Python (like the `random` library mentioned in a previous post). These libraries can do anything from machine learning to date and time formatting to meme generation. If you have a task you need done, most likely someone has written a library for it.

There are three main things to keep in mind when using a library:

1. You need to install it. Installation in GNU/Linux based systems will generally be easier than on Windows or OSX, but there will always be documentation for how to do it.
2. You need to import it. At the top of your program, make sure you write the line `import requests`, or whatever the name of your library is. Then you can use it to your heart's content.
3. You need to read documentation. Someone else wrote it, so the rules might not be so obvious. Anyone (or any group) that writes a Python package writes documentation for it. Eventually, reading documentation will become second nature.

Requests

One of the most useful libraries written for Python recently, requests does "HTTP for humans." What this means in laymen's terms is that it asks the internet things from Python. When you type "facebook.com" into the browser, you are asking the internet to show you Facebook's homepage.

In the same way, a program can ask the internet something. It might not be "show me Facebook", but you can for example ask Github for a list of all the repositories that the user "mprat" has. You can do this with an API (Application Programming Interface). This exercise doesn't use APIs, so we'll talk more about those in a later post.

Back to showing the user a webpage. When I type “facebook.com” into the browser, Facebook sends my browser a bunch of HTML (basically, code for how the website looks). The browser then takes this HTML and shows it to me in a pretty way. (Fun fact: to see the HTML of any page in a browser, right click on the page and “Inspect Element” or “View Source” depending on your browser. In Chrome, “Inspect Element” will pop up a module at the bottom of your page where you can see the HTML from the page. This trick will come in handy when you’re doing the exercise. If you need to DO anything with this HTML, better to use a program. More posts about this coming later.) If I want to “see” a webpage with a program, all I need to do is ask it for it’s HTML and read it.

The ‘requests’ library does half of that job: it asks (requests, if you will) a server for information. This could be just data (through an API - more later) or in the case of this exercise, HTML.

Look at the documentation for all the details you need. In this particular latest version, all you need to do to ask a website for it’s HTML is:

```
import requests
url = 'http://github.com'
r = requests.get(url)
r_html = r.text
```

Now inside the variable `r_html`, you have the HTML of the page as a string. Reading (otherwise called **parsing**) happens with a different Python package.

BeautifulSoup

To solve our problem of parsing (reading, understanding, interpreting) the string of HTML we got from `requests`, we use the `BeautifulSoup` library.

What it does is give a *hierarchical* (a pyramid structure) to the HTML in the document. If you don’t know anything about HTML, the Wikipedia article is a good summary. For the purposes of this exercise, you don’t need to know anything about HTML beyond being able to look at it quickly.

Because `BeautifulSoup` takes care of interpreting our HTML for us, we can ask it things like: “give me all the lines with `<p>` tags” or “find me the parent element to the `<title>` element”, etc.

Your code would look something like this:

```
from bs4 import BeautifulSoup

# some requests code here for getting r_html

soup = BeautifulSoup(r_html)
title = soup.find('span', 'articletitle').string
```