**Introduction**

***Web Scraping***

The automated gathering of data from the Internet is nearly as old as the Internet itself. [Ryan Mitchell]. Scraping is a process by which you can extract data from an html page, into a CSV or other format, so you can work with in Excel and use in your visualizations.

**urllib** is a standard Python library (meaning you don’t have to install anything extra to run this example) and contains functions for requesting data across the web, handling cookies, and even changing metadata such as headers and your user agent.

**urlopen** is used to open a remote object across a network and read it. Because it is a fairly generic library (it can read HTML files, image files, or any other file stream with ease),

The **BeautifulSoup** library was named after a Lewis Carroll poem of the same name in Alice’s Adventures in Wonderland. It helps format and organize the messy web by fixing bad HTML and presenting us with easily-traversible Python objects representing XML structures.

***Regular Expressions***

They are so called regular expressions because they are used to identify regular strings: Regular string is any string that can be generated by a series of linear rules. The example this linear rules can be in identifying an email address. Below are the rules:

Rule 1: **[A-Za-z0-9\. +]+**

The first part of an email address contains at least one of the following:

uppercase letters, lowercase letters, the numbers 0-9, periods (.), plus signs (+), or underscores (\_).

Rule 2: **@**

After this, the email address contains the @ symbol.

Rule 3: **[A-Za-z]+**

The email address then must contain at least one uppercase or lowercase letter.

Rule 4: **\.**

The email address then must contain at least one uppercase or lowercase letter.

Rule 5: **(com|org|edu|net)**

Finally, the email address ends with com, org, edu, or net (in reality, there are many possible toplevel domains, but, these four should suffice for the sake of example).

Although there are a lot of ways to do web scraping, this handout will focus on discussing making a Python program that gets data from the web.

* Part 1: Basic Web Scraping with Beautiiful Soup
* Part 2: Use Regular Expression in Web Scraping

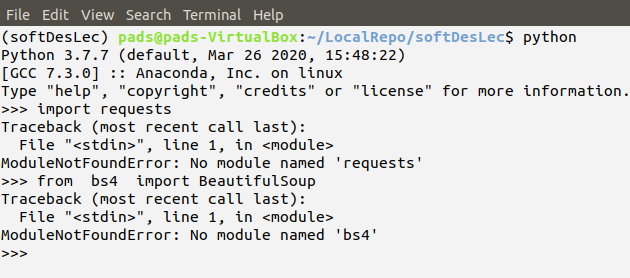
See below **OneDrive** links for your guide**.**

* Screengrabs: <https://tinyurl.com/teclnqy>
* Python Files: <https://tinyurl.com/rmfrycm/>
* Cheatsheets: <https://tinyurl.com/ufd7y2q>

1. **Web Scraping with Beautiiful Soup**
2. Preparation
3. Create the softDesLec environment using the following conda command below.

**conda create --name softDesLec python=3.7**

Activate the softDesLec and run the python statement in the python interpreter. You should see the same error as shown in the below screengrab.

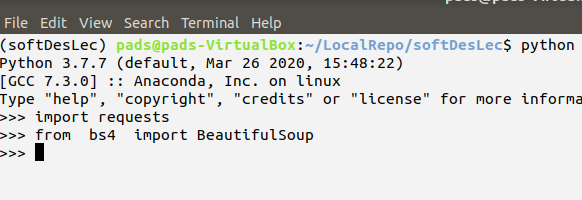


1. Exit python interpreter and install requests and bs4 modules using the below pip install commands.

**pip install requests**

**pip install beautifulsoup4**

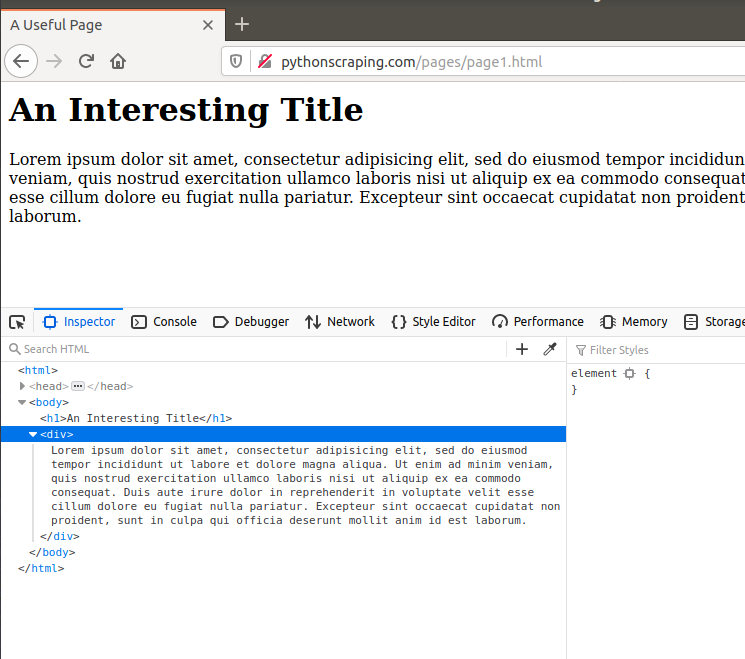
You will see that the python interpreter will now recognize the two python statements as shown below.



1. You may now use Visual Studio Code for the sample code below. You need to choose the softDesLec anaconda environment before running the scrapetest.py. Refer to the VS Code cheatsheet guide pdf file on how to activate the environment. The cheatsheet provides the short-cut key for running the file in the VS Code. You may also choose to run the script in the Ubuntu Linux terminal or Anaconda terminal as shoen in screengrab (b).

|  |  |
| --- | --- |
|  |  |
| 1. Running scrapetest.py in Visual Studio Code | 1. Running scrapetest.py in Ubuntu terminal |

1. Open a web browser (Firefox or Google Chrome) and examine the url in the scrapetest.py program. See below screengrab.



1. Creating an instance of the BeautifulSoup Class

The two lines below are needed to create the instance of the BeautifulSoup class.

**from bs4 import BeautifulSoup**

**soup = BeautifulSoup(html\_doc, 'html.parser')**

1. BeautifulSoup find() and findAll() Functions

Syntax:

**findAll(tag, attributes, recursive, text, limit, keywords)**

**find(tag, attributes, recursive, text, keywords)**

1. **Use Regular Expression and Beautiful Soup**
2. Open the below web page take time to examine the page source as shown below

<http://www.pythonscraping.com/pages/page3.html>

|  |  |
| --- | --- |
|  |  |
| 1. Inspecting a page element | 1. Viewing the img equivalent html tag |

The product images on the page has the following form:

<img src="../img/gifts/img3.jpg">

Then, the below web scraping python code can be used to get all the img tag and src. You can download the **regEx\_product\_img.py** in the given OneDrive link. You should have the same display as shown in screengrab (b) below.

|  |  |
| --- | --- |
|  |  |
| 1. Python file to get all the product images | 1. Sample run of the code given in (a) |

The regular expression provided in the compile function is this:

**"\.\.\/img\/gifts/img.\*\.jpg"**

**References**:

Books

• Web Scraping with Python

Ryan Mitchell

Free Online Sources:

• https://blog.hartleybrody.com/web-scraping-cheat-sheet/

• https://realpython.com/python-requests/