**CS506 Programming for Computing**

**HOP05A Web Scraping**

04/10/2020 Developed by Amrutha Vaidyanathan and Apiwat Chuaphan

10/01/2020 Reviewed by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Create python programs to get information from the web
* Automate your programs to control websites
* Read and write spreadsheets using Python

**Resources**

* [Automate the Boring Stuff with Python](https://login.proxy.cityu.edu/sso/skillport?context=89288)

**Preparation**

1. In Visual Studio Code, open the private repository generated when you accepted the HOP05 assignment (If you cannot find that repository in your machine, you might have not cloned the repo, if so, please do before proceeding).

A screenshot of a cell phone

Description automatically generated

**Web Scraping**

The term for using a program to extract and process data for the Web. In this Module, we will cover Python Modules including:

***Webbrowser*** *—* comes with Python and opens a browser to a specific page.

***Requests*** *—* downloads files and web pages from the Internet.

***Beautiful******Soup*** *—* parses HTML.

***Selenium*** *—* launches and controls a web browser.

1. We will start with **webbrowser** module and create a file called **search\_map.py**and type the code below.

A screenshot of a cell phone

Description automatically generated

The webbrowser module includes functions to open URLs in interactive browser applications. The sys module is included in the standard libraries and contains the functions and other data necessary for your code to perform introspection about the system in which its running. With the len(sys.argv) function you can count the number of arguments.

In the terminal type the following.

>>> python3 search\_map.py city university of seattle

**Note:** If you are getting 404 error replace the existing link with <https://google.com/maps/>

Type **city university of Seattle** as an argument and it will bring you to Google map and search for the place. You can also try giving different places of your choice.

A close up of a map

Description automatically generated

1. Moving on to the next module, **Request**, it lets you download files from web. It does not come with Python, so you need to install it first. Run the following command in your terminal.

>>> **pip3 install requests**

Create **request.py** and type the following. If requests module has been installed correctly, no error message will show up.

A screenshot of a cell phone

Description automatically generated

The raise\_for\_status() returns an HTTP Error object if an error has occurred during the process. It is used for debugging the requests module and is an integral part of Python requests. Python requests are generally used to fetch the content from a particular resource URI.

The status\_code returns a number that indicates the status (200 is OK, 404 is Not Found). Python requests are generally used to fetch the content from a particular resource URI. Whenever we make a request to a specified URI through Python, it returns a response object.

>>> In the terminal type **python3 request.py**

You will see the text content printed out to the terminal along with status code and the size like this if the URL is valid.

A picture containing drawing

Description automatically generated

Notice that the program will genarate zero error since the URL is valid at the time this program was written, if the URL is invalid it will generate 404 client error.

1. Parsing HTML with **BeautifulSoup** module (for extracting information from HTML page). It does not come with Python. To install, run this command in the terminal.

>>> **pip3 install beautifulsoup4**

Let’s extract information from local HTML file: first create **example.html**and copy/paste the text below.

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<title>This is Title</title>

</head>

<body>

<p>

Read the Doc

<strong>

BeautifulSoup

</strong>

from

<a *href*="https://www.crummy.com/software/BeautifulSoup/bs4/doc/">Documentation</a>.

</p>

<p *class*="slogan">Scrapping the easy way!</p>

<p>By <span *id*="author">Elliot Alderson</span></p>

</body>

</html>

Next, create **beautifulsoup.py**and type the following.

A picture containing drawing

Description automatically generated

An HTMLParser instance is fed HTML data and calls handler methods when start tags, end tags, text, comments, and other markup elements are encountered. The select fetches mentioned attribute. In the given code # depicts the id.

>>> In the terminal type **python3 beautifulsoup.py**



Output would be the string inside id=”author” tag, which is Elliot Alderson

1. This step we will try to extract information from real website. Add the following to **beautifulsoup.py**

A screenshot of a cell phone

Description automatically generated

Look at the output to understand the difference between printing the element and getting only the text of the element and printing it.

>>> In the terminal type **python3 beautifulsoup.py**

A screenshot of a cell phone

Description automatically generated

**Note:** if you’d like to see all HTML text, you could **print(soup.prettify())**

Why html.parser? This [link](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-a-parser) might be helpful

1. Controlling the browser with **Selenium** module that lets Python to click links and fill in the login information fields. To install, type this command in the terminal.

>>> **pip3 install selenium**

Selenium is a tool to test your web application. You can do this in various ways, for instance

* Permit it to tap on buttons
* Enter content in structures
* Skim your site to check whether everything is "OK" and so on.

To start a web browser, the Selenium module needs a web driver. Python interacts with the selenium web driver and the web driver interacts with the browser.

**Download drivers for your browser.**

Chrome - <https://sites.google.com/a/chromium.org/chromedriver/downloads>

Firefox - <https://github.com/mozilla/geckodriver/releases>

Edge - <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

Safari - <https://webkit.org/blog/6900/webdriver-support-in-safari-10/>

Note: Find the version of your browser and download the driver accordingly.

Make sure it’s in your PATH, (e. g., place it in /usr/bin or /usr/local/bin.)

I use mac and had the chrome driver in the download folder. So, used the following command for copying the driver file into the **/usr/local/bin**.



Note: My driver file is **chromedriver**. Make sure you copy the correct file.

If using Windows, you can drag and drop the file.

1. Let’s make a program **automate\_web.py** to automatically fill in a form for you. Type the code in the picture below and run it.

A screenshot of a cell phone

Description automatically generated

>>> In the terminal type **python3 automate\_web.py**. The browser will open displaying the following page and automatically fill in the information for you.

**Note:** Mac user might be blocked from running the program

**Solution:** On your Mac, choose Apple menu --> System Preferences --> click Security & Privacy --> click General --> click "Allow Anyway"

A screenshot of a cell phone

Description automatically generated

Learn more about Selenium >>> <https://selenium-python.readthedocs.io/>

**Excel Spreadsheets**

The openpyxl module allows Python programs to read and modify spreadsheets. To install, type this command in the terminal

**>>> pip3 install openpyxl**

1. Download **example.xlsx**(available under Module 5 folder you cloned).
2. Create a file **read\_excel.py**and type the following.

A screenshot of a cell phone

Description automatically generated

openpyxl.load\_workbook() is used to open an existing workbook

>>> In the terminal type **python3 read\_excel.py**.

A close up of text on a black background

Description automatically generated

1. Now we will write some to a excel file. Create **write\_excel.py** with the following.

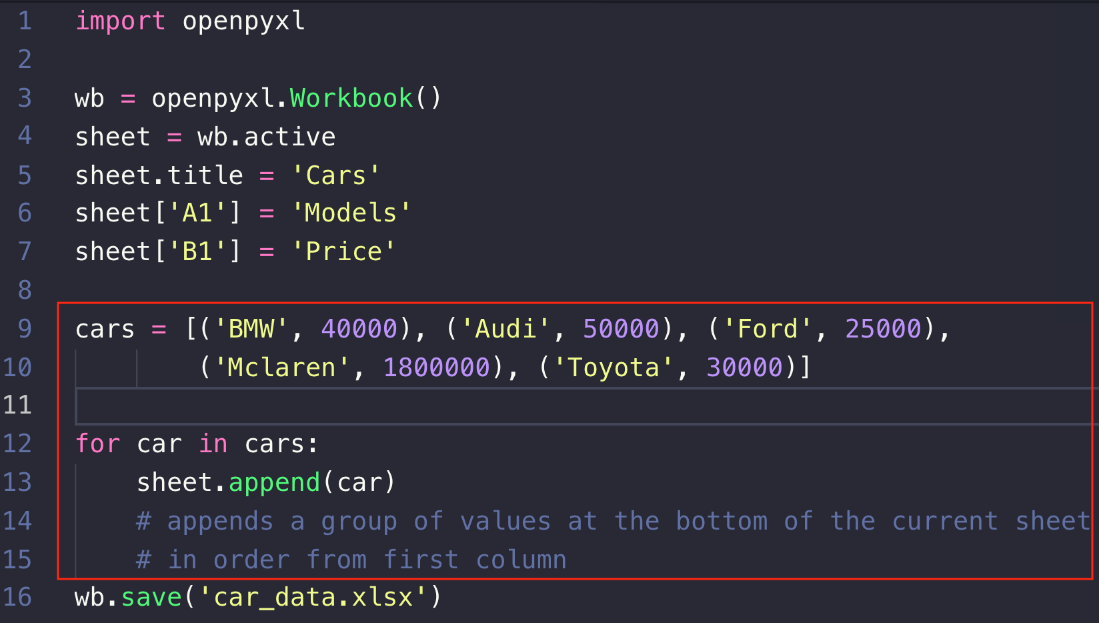
A close up of text on a black background

Description automatically generated

>>> In the terminal type **python3 write\_excel.py**.

You will get Excel file named “car\_data.xlsx” with 2 data.

1. Let’s append more data on the Excel file by updating it as the below screenshot. Use the same python file from the above step.



Check the Excel file and see the new data has been added. More info on openpyxl on this [link](https://openpyxl.readthedocs.io/en/stable/api/openpyxl.worksheet.worksheet.html#openpyxl.worksheet.worksheet.Worksheet.iter_cols)

**Push your work to GitHub**

Open the terminal from the VSCode by hitting the “control” + “~” key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 5 – Your Name”

>>> git push origin master