**CS506 Programming for Computing**

**HOS03 Web Scraping and Working with Spreadsheets**

1/31/2025 Reviewed by Jonathan Koerber

3/13/2025 Reviewed by Clark Ngo

4/17/2025 Reviewed by Naveena Moddu

04/17/2025 Reviewed by Ashwin Paturi

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

**Resources**

* Sweigart, A. (2020). *Automate the boring stuff with Python: practical programming for total beginners* (2nd ed.). No Starch Press.

**Learning Outcomes**

* **Section 1: Preparing your environment**
* **Section 2: Web Scraping**
* **Section 3: Excel Spreadsheets**

**Section 1: Preparing your environment**

Get started with your virtual environment here: <https://cityuseattle.github.io/docs/git/github_codepsace/#codespaces>

**Section 2: 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 the **webbrowser** module, create a file called **search\_map.py,**and type the code below.

A computer screen with text on it

Description automatically generated

The web browser 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 it's running. You can count the number of arguments with the len(sys.argv) function.

In the terminal, type the following.

**>>> python search\_map.py city university of seattle**

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

Type the **City University of Seattle** as an argument, and it will bring you to Google Maps, where you can search for the place. You can also try giving different locations of your choice.

A close up of a map

Description automatically generated

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

>>> **pip3 install requests**

Create **request.py** and type the following. If the requests module is installed correctly, no error message will appear.

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 generally 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 generally fetch the content from a particular resource URI. Whenever we request a specified URI through Python, it returns a response object.

In the terminal, type the following

>>> **python request.py**

You will see the text content printed out to the terminal along with the 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 the **BeautifulSoup** module (for extracting information from HTML pages). This module does not come with Python. To install it, run this command in the terminal.

>>> **pip3 install beautifulsoup4**

Let’s extract information from the 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 the mentioned attribute. In the given code, # depicts the ID.

In the terminal, type the following

>>> **python beautifulsoup.py**



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

1. This step we will try to extract information from real websites. 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 “python 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

For this mini section, we can only experience Selenium in our local machine.

1. Install Python in your machine <https://cityuseattle.github.io/docs/environment/python/>
2. Install VS Code <https://cityuseattle.github.io/docs/tools/vscode/>
3. 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://developer.chrome.com/docs/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.**

**[Find Your Chrome Browser Version: Open Google Chrome. Click on the three dots in the upper-right corner to open the menu. Navigate to Help > About Google Chrome. You will see your browser's version number (e.g., Version 116.0.5845.140)]**

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

For Mac Users who have the Chrome driver in the download folder, use the following command to copy the driver file into the **/usr/local/bin**.



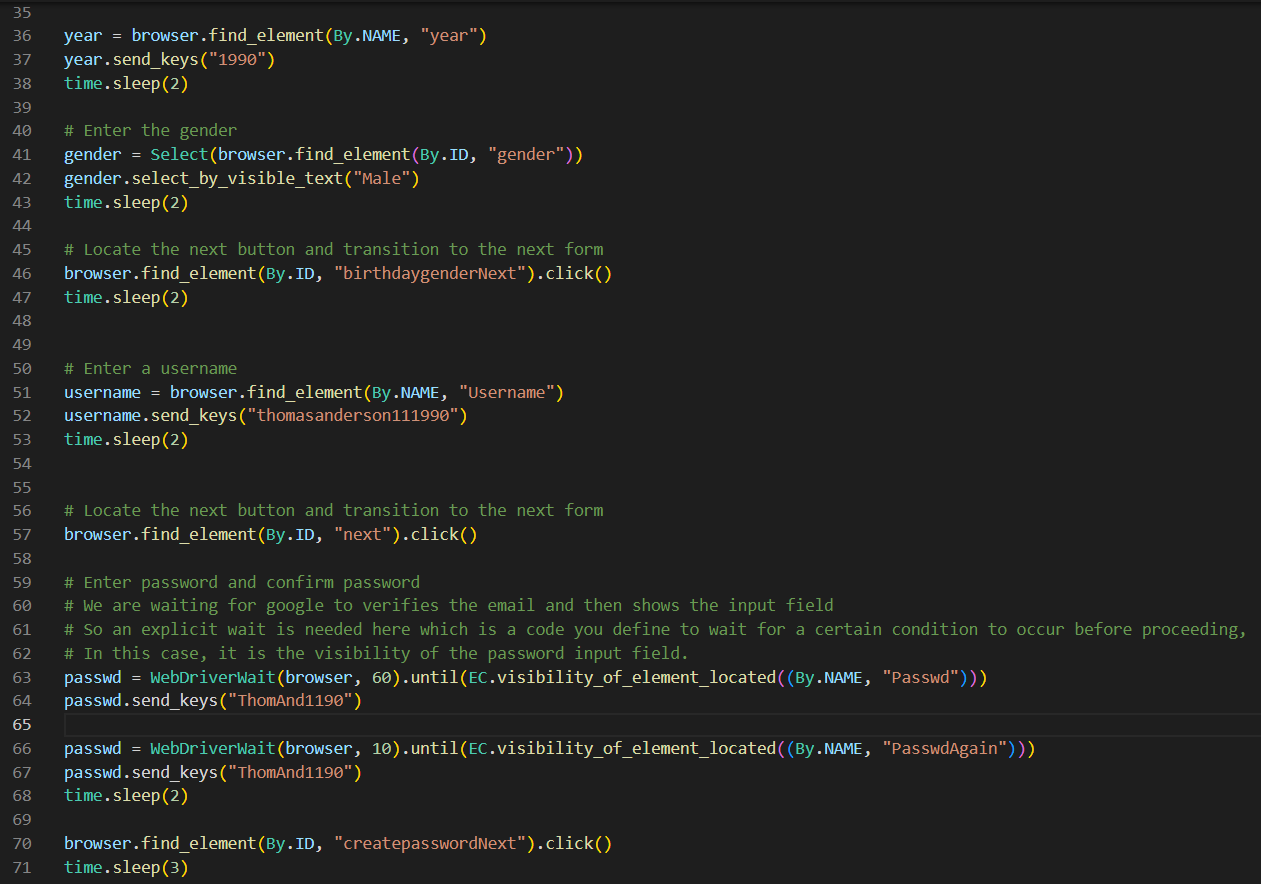
Note: The driver file in the screenshot above is **Chromedriver**. Make sure you copy the correct file.

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

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

A computer screen shot of text

Description automatically generated

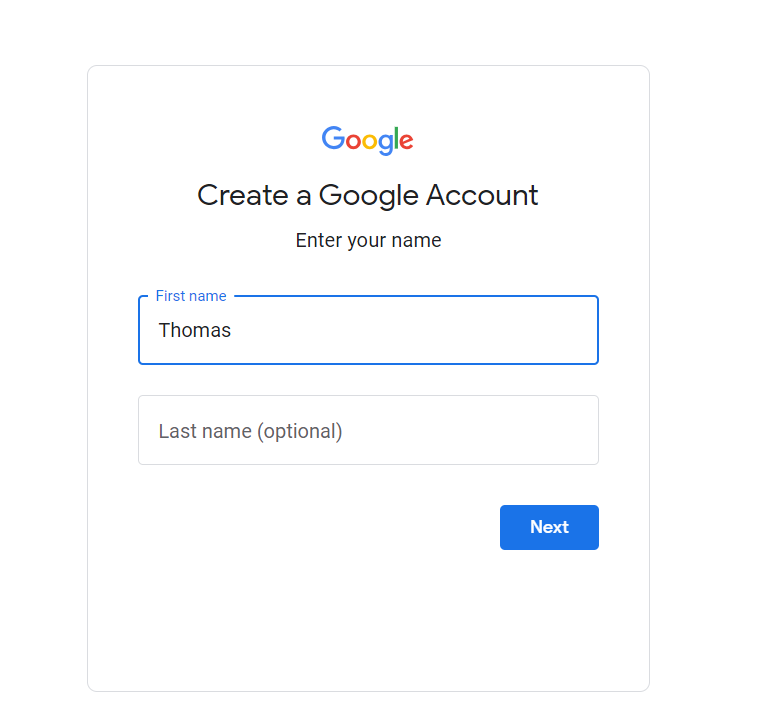


>>> In the terminal, type **“python automate\_web.py**.”

The browser will open, display the following page, and automatically fill in the information for you.

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

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



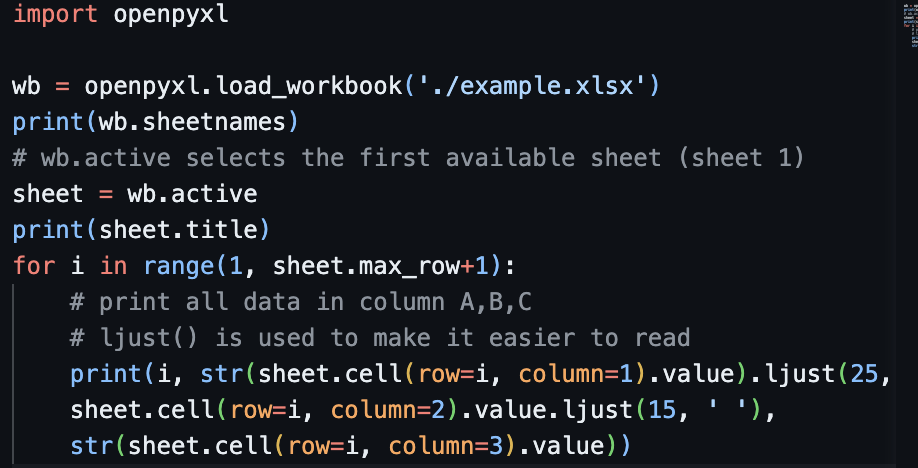
Learn more about Selenium >>> <https://selenium-python.readthedocs.io/>

**Section 3:** **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. Verify that **example.xlsx**is available.
2. Create a file **read\_excel.py**and type the following:



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

>>> In the terminal type “**python read\_excel.py”**

A close up of text on a black background

Description automatically generated

1. Now, we will write some in an Excel file. Create **write\_excel.py** with the following.

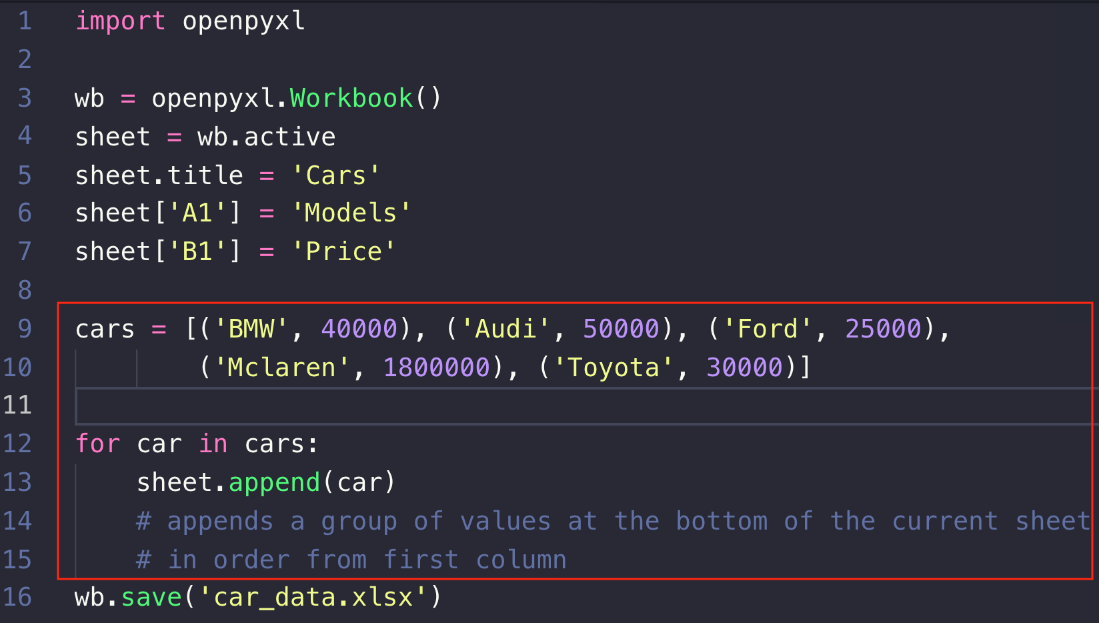
A close up of text on a black background

Description automatically generated

>>> In the terminal, type “**python write\_excel.py”**

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

1. Let’s update the Excel file to append more data, as shown in the screenshot below. 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 Codespaces**

Follow instructions here: <https://cityuseattle.github.io/docs/git/codespaces_submission/>