Accessing Website Content:
Web Harvesting 101 with Python on KLC

Disclaimer: *This primer assumes you have basic familiarity with web scraping tools and python.*

If you found a website with information you would like to extract, web scraping offers a mechanism to automate a solution. For research purposes, web scraping can be broken down into two steps:

1. Accessing and saving the website content
2. Parsing the content for the data you require

Web Scraping Checklist

Before you "scrape" a website, consider the following checklist:

- ✓ Does the website offer a direct data download or an API for you to use?
- ✓ Does Northwestern already subscribe to this website?
- ✓ Does using an automated user agent on the website violate the **Terms of Service** or **robots.txt** file of the site?
- ✓ Do you need to misrepresent yourself to access the website's content (i.e., create a fake account, spoof your user agent or IP address)

If you answered **YES** to any of these questions, then **DO NOT SCRAPE** the website.

Advantages of Using KLC

If you cleared the checklist above and can scrape a website, the Kellogg Linux Cluster offers a number of advantages during different steps in the scraping process:

1. You do not have to load any software directly on your computer to scrape.
2. You can start your scraper on the server and walk away as it continues to run.
3. You can save the scraping output of up to 80GB in your home directory
4. While you will want to limit your direct requests/hits to a website to prevent getting blocked or slowing down a website's server (*see **time** library in Python below*), you can take advantage of your 8-core KLC limit by parsing the html pages you download in parallel.

The Setup and some Sample Code

Once you log in to a terminal window on KLC (possibly through FastX GNOME terminal or your computer's terminal), creating a conda environment is an easy way to install and load the appropriate software on KLC without worrying about dependencies. NU IT provides a valuable guide for using Python on Quest/KLC here.

To access python on KLC, you will use the anaconda module. From the command line type:

```
module load python/anaconda3.6
```

Creating a conda environment

We recommend installing the python libraries you need through creating a conda environment. You can create one by typing the following from the command line:

```
conda create –n <your environment’s name> python=3.6
conda install –c <any packages you need>
```

For example:

```
conda create –n scrape_env python=3.6
conda install -c conda-forge requests # for the Requests library
conda install -c conda-forge beautifulsoup4 # for Beautiful Soup
conda install –c conda-forge selenium # for Selenium
```

Note that you will only need to create your python environment once. Every subsequent time you want to access the conda environment simply load anaconda and activate the environment.

```
module load python/anaconda3.6
source activate scrape_env
```

Running the requests and Beautiful Soup libraries

For the **requests** and **Beautiful Soup** packages, you will not need to load any other software. Here is a sample python file that utilizes both packages to scrape the faculty member index of the Kellogg website.

```
# Scraping Example 1 – scrape1.py

# libraries to import
import bs4 as bs
import urllib.request
import time

# identify and load a website
url = ‘https://www.kellogg.northwestern.edu/faculty/faculty_directory.aspx’
source = urllib.request.urlopen(url).read()

# create a soup
soup = bs.BeautifulSoup(source, ‘html.parser’)

# find the first faculty member’s website
faculty = soup.find(‘div’, {‘id’: ‘bindFaculty’})
profs = faculty.findAll(‘h2’,{‘id’: ‘facName’})
website = profs[0].find(‘a’, href=True)
print(website[‘href’])
```

Save a copy of this file (**scrape1.py**) to KLC. To execute python, type the following:

```
python scrape1.py
```

Running Selenium

In order to launch selenium on KLC, you will also need access to a web browser. To use Firefox, please type the following in the command line:

```
export PATH=/kellogg/bin:$PATH
module load firefox/62
```

Here is a sample python file that utilizes Selenium on the faculty member index of the Kellogg website.

```
# Scraping Example 2 – scrape2.py

# libraries to import
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.firefox.options import Options
options = Options()
# options.set_headless(headless=True) # to run selenium without opening a web
browser
import time

# identify and access a website
url = ‘https://www.kellogg.northwestern.edu/faculty/faculty_directory.aspx’
driver = webdriver.Firefox(firefox_options=options)
driver.implicitly_wait(3) # wait 3 seconds before accessing website
driver.get(url)

# click open the “More Faculty” button
python_button = driver.find_element_by_link_text(‘MORE FACULTY’)
python_button.click()

# sleep times
time.sleep(10)

# end browser session
```

Save a copy of this file (**scrape2.py**) to KLC. To execute python, type the following:

```
python scrape2.py
```

Running a Jupyter Notebook

NU IT also provides guidance on how to launch a Jupyter Notebook on Quest/KLC here. From the command line select a browser and type the following:

```
module load firefox/62
jupyter notebook --browser=firefox
```