Assignment 2.0 - Scraping the Web

Overview

This week, you will be scraping Wikipedia and storing information about actors and movies into a data structure of your design. You will also write a graph library and a function for converting your data to a graph. Then you will store the relevant information as a JSON file so you can load it again without re-scraping the website. Finally, you should be able to provide basic information from your data structure through console output.

> **Programming Language**
> Unlike past weeks, this week, you should implement your project in a programming language which you have not used. For more information, see #Programming Language Selection.

Motivation and Goals

There are many methods of data collection in the rapidly-evolving world of information and technology, but web scraping is among the most popular and accurate. In layman's terms, web scraping is the act of using bots to extract specific content and data from a website. Web scraping is especially useful because it has the ability to convert non-tabular, nonsensical and poorly constructed data into something both in format and in content. Web scraping is also championed for its ability to acquire previously-inaccessible data. However, web-scraping is not about mere acquisition-- it can also assist you to track changes, analyze trends and keep tabs on certain patterns in specific fields.

The purpose of this particular assignment is to introduce you to the real-world application of web-scraping tech, as well as get you thinking about the creative process that accompanies the tasks you are assigned. There will be a number of directives that you will have to solve both in this assignment as well as when you graduate and break into industry-standard workplaces, so keep this in mind as you work on this assignment. Web scraping may be the focus of this particular assignment, but it very well may be a potential, real-life approach you use in the future.

For this practice assignment, we will be using Wikipedia as our web source, for a number of reasons. Although Wikipedia provides database dumps for everything, it is the best source to use for this exercise because not only does it have fairly up-to-date information, it is also legal to scrape Wikipedia without ramifications or complicated restrictions.

Programming Language Selection

Whatever language you choose, you should use an IDE of your choice (suggestions of Ruby & Python below):

- Python Consider using PyDev for Eclipse or PyCharm (from the makers of IntelliJ)
- Ruby Consider using a plugin for eclipse or RubyMine (from the makers of IntellIJ)
- Javascript

You can also select a language you would like to learn (ideally something not too obscure), and contact your moderator or the TAs to ensure that this language is appropriate to use.

> **Language Selection**

Be aware that the TAs are not familiar with every programming language out there, so we may or may not be able to provide in-depth help with the language you choose. If we cannot help you, post your question to Piazza since there may be other people in the course that can help with your problem.

Background

Logging

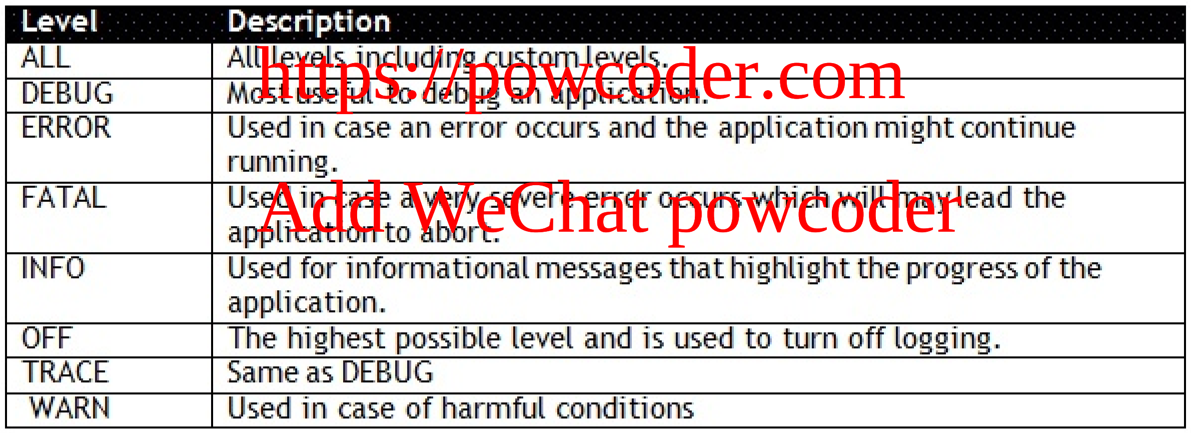
We are asking you to perform logging as part of your web scraping code.

What is logging?

> When a function is called, I log that it has been called along with any passed in arguments, the time it was called, and in Javascript, the name of the function that called it. I also log before a function returns, along with the time and any relevant return data. I log when asynchronous calls return. I log when significant logic changes happen. -Erik Hazzard, How Logging Made me a Better Developer

In short, logging is adding output or log statements to your code that indicates the status of what is going on while your code is running. Log statements are helpful because they decrease code complexity, help debugging, increase visibility and assist communication with other developers.

There are several different log levels.



Level	Description
ALL	All levels including custom levels.
DEBUG	Most useful to debug an application.
ERROR	Used in case an error occurs and the application might continue running.
FATAL	Used in case a very severe error occurs which will may lead the application to abort.
INFO	Used for informational messages that highlight the progress of the application.
OFF	The highest possible level and is used to turn off logging.
TRACE	Same as DEBUG
WARN	Used in case of harmful conditions

You are not required to use all of these log levels when writing your code, but we expect you to use at least 3 of the available log levels in the language you are using. For example, for Python you could log a WARNING level log when you encounter a strangely formatted page that cannot be understood by your scraper. In addition, we expect you to write *meaningful* log messages. In other words, your log message should not be "logging.warning('Some error')" or "logging.warning(i)"; if we see vague or meaningless messages like this we will take off points. Using the above example, an acceptable message could be "logging.warning('[page] was not able to be parsed because filmography section was not found.')". Since the pages you are scraping will not be uniform in their HTML markup, it is very likely that you will encounter errors and edge cases as you scrape. Take advantage of logging for these cases!

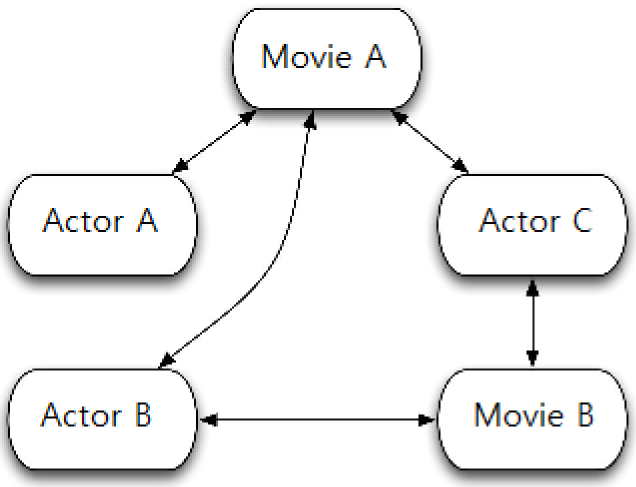
Here are some resources to get you started:

- Python: Python 2 Logging, Python 3 Logging
- Ruby: Logging with Ruby
- Javascript: Logging with Javascript

You are free to use any library you like for logging in addition to those suggested above.

Graphs

A graph, at least to computer scientists, is a set of vertices and a set of edges that connect them. Each vertex in a (directed) graph has a set of outgoing edges and a set of incoming edges. There are many ways to represent a graph in memory, it is up to you to determine which representation you would like to use. See the image below for a graphical representation of what your graph might look like:



For this assignment, you should be able to represent each actor and movie as a vertex in your graph. The weight of an edge should represent the amount of money the actor made from the movie. However, since this information is not readily available on the Wikipedia page, we leave the choice of how to represent this up to you. You should **not** assign the same weight to all your edges for a given vertex. Here are some examples of what you can do:

- Represent the weight of an edge as the grossing value of the movie, with actors at the top of the cast list having more weight than those at the bottom of the list
- Represent the weight of an edge as the grossing value of the movie, with young actors being weighed more than older actors
- Represent the weight of an edge as the grossing value of the movie, with actors who have been in more movies having more weight than those who have been in fewer movies.

Also, your graph will need to be augmented to hold extra data about each movie. You will have to adapt your implementation to handle this extra data. Note that in our case the graph is undirected because if an actor is in the movie then the movie definitely contains that actor.

JSON

We are asking you to store your scraped data as a JSON file.

What is JSON?

> JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. - json.org

JSON is most commonly used in web applications, as a lightweight format to transfer data asynchronously between a client and server. For instance, Facebook transmits status updates to your newsfeed using JSON so that new posts appear without having to reload the page. But, JSON is not limited to web applications. Parsers for JSON are available for just about every language out there. See http://www.json.org for parsers for some of the most common programming languages. It would be wise for you to pick a language for this assignment that already has well developed parser. For instance, JSON parsing is built into Python's core library.

You can read more about JSON on Wikipedia.

Part 0: Reading

Before you begin web scraping, make sure to read the following links:

- https://www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/
- https://www.promptcloud.com/blog/dont-get-blacklisted-legitimate-web-scraping-process
- https://www.promptcloud.com/blog/some-traps-to-avoid-in-web-scraping/

To avoid harming the website, please avoid extensive scraping. Be a responsible scraper! 🙂

Part I: Web Scraping

Here's what you will need to do:

- Write a web scraper to gather information from pages on Wikipedia. You should gather information from a large amount of movies page (>125) and actor pages (>250). The choice of the starting page is up to you, but feel free to start here: https://en.wikipedia.org/wiki/Morgan_Freeman
- Store grossing information for movies and age for actors.
- Log messages to a log file as you scrape. You should log at least three different levels of messages. Messages should be *appropriate* and *meaningful*.

You **can not** use the Wikipedia API; however, feel free to use whatever scraping API your language supports. Here are some of our recommended libraries:

- Python: BeautifulSoup (Documentation), Scrapy
- Ruby: Nokogiri
- Javascript: Node.js, Scraping with Node.js

Your web scraper should not run for longer than 10 minutes with the requirements we have given you. For your live testing in section, if your scraper takes longer than your time slot (20 minutes in a 6 person section), you will not receive credit for that part of the rubric!

Part II: Graph Library

Next, you will need to create a general graph library for your application.

You **can not** use any existing graph library to handle the representation of your graph in memory. That is, you need to **write your own** graph implementation for this assignment. For this section, implement a general graph library that will hold the data you parse and interact with in the following sections.

Testing

Just as with past assignments, you should test your code thoroughly using unit tests. Your tests should be automated, and perform asserts similar to jUnit, using a unit test library available in your language of choice. Contact the course staff or post on Piazza if you have trouble finding a test suite in your language.

You are required to be able to convert your web scraped data into a graph structure. This will be important for next week. However, how you choose to represent your web scraped data in order to do the following parts (III and IV) is up to your choice.

Part III : JSON Storage and Retrieval

You should be able to

1. Store your data structure as a JSON file
2. Load your data structure from a JSON file

You **can** use any existing JSON library you wish for this task.

Part IV : Graph Queries

Your code should have methods that allow the user to:

1. Find how much a movie has grossed
2. List which movies an actor has worked in
3. List which actors worked in a movie
4. List the top X actors with the most total grossing value
5. List the oldest X actors
6. List all the movies for a given year
7. List all the actors for a given year

All of this information should be calculated at query time and not be hard-coded into your source. Be prepared to run these methods and produce console output for your moderator in section.

Testing

As usual, we require that you write extensive unit tests for each part of this assignment. We understand that it can be difficult to test for web scraping. However, make sure to exhaustively test all other parts of your code. If your language does not have a testing framework (unlikely), you will need to implement your own test runner and utilities to accomplish this part of the assignment. In order to test your web scraper, your moderator will ask you to scrape an actor/movie page of their choice in section, so **be prepared**.

- Summary -

Table of Contents

Reading

- Code Complete Chapter 7: High-Quality Routines
- Optional: The Pragmatic Programmer Chapter 5-26: Decoupling and the Law of Demeter
  - ➔ The readings are due before lectures every Friday.

Submission

This assignment is due at the **beginning of your discussion section the week of October 8th, 2018**. Please be sure to submit in GitLab, and ask your moderator or TA before the deadline if you have any questions.

Objectives

- Learn about responsible web scraping
- Learn a new programming language
- Learn about graphs and how to implement them
- Learn to work with data in JSON format
- Learn how to effectively decompose your code into reusable modules

Resources

- Python
- Ruby
- Javascript
- JSON Homepage
- JSON Wikipedia
- Graphs

Grading

Category	Weight	Scoring	Notes
Basic Preparation	2	0-1	Ready to go at the start of section
Cleverness	2	0-2	The hardest points on the rubric
Code Submission	4	0-2	Submitted on time and to the correct location in the repository
Decomposition	4	0-2	Reference Wiki Grading Page for more information
Documentation	4	0-2	Reference Wiki Grading Page for more information
Effort	2	0-2	Reference Wiki Grading Page for more information
Naming	2	0-2	Reference Wiki Grading Page for more information
Overall Design	5	0-2.5	Reference Wiki Grading Page for more information
Participation	5	0-2.5	Reference Wiki Grading Page for more information
Presentation	4	0-2	Reference Wiki Grading Page for more information
Requirements - Web scraping and queries	5	0-2.5	• 0 points: Lacks any form of a Web Scraper, or utilizes non-scraping library to fetch information • 1 points: Has a semi-functional Web Scraper (may crash on certain web pages, or fails to reach threshold), lacks full querying functionality • 2 points: Functional Scraper reaches the minimum threshold, allows for simple queries • 2.5 points: Scraper is fully functional and covers the various webpage designs, can

			process advanced queries (beyond what is mentioned on the wiki)
Requirements - Logging	2	0-1	• 0 points: Lacks Logger or logger does not provide relevant details • 1 point: Logger uses tags to id messages and provides relevant information to understand programing execution
Requirements - Programming Language	2	0-1	Using an appropriate programming language.
Requirements - Graph Library	2	0-1	• 0 points: Data Structures hold no semblance to a graph library • 1 point: Data Structures follow some graph library design, and allow for easy graph traversal functionality
Requirements -JSON storage and retrieval	2	0-1	• 0 points: No JSON storage/retrieval • 1 point: Functional JSON storage/retrieval
Testing - Graph Construction	4	0-2	• 0 points: No unit tests • 1 point: Sub 80% code coverage or unit tests fail to cover graph related functionality • 2 points: 80%+ code coverage on data structures, unit tests cover graph related functionality
Testing - Live Testing	4	0-2	• 0 points: Scraper fails to grab any pertinent information or fails immediately • 1 point: Scraper fails to reach the required limit • 2 points: Scraper succeeds in scraping enough information to reach the specified limits
Total	**55**		