Web **S**craping using **P**ython**3**

A Project Report for Summer Industrial Training

Submitted by

Mayuraksha Sikdar

in partial fulfillment for the award of the degree of

B. Tech

in

Computer Science & Engineering

Narula Institute of Technology (affiliated to MAKAUT)

At

Udemy.

Udemy, Inc.

COMPLETION CERTIFICATE

Name of Instructor: Colt Steele

ACKNOWLEDGEMENT

I take this opportunity to express my deep gratitude and sincerest thank to my course instructor, **Colt Steele** for giving most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

1. ABSTRACT:

Web Scraping is a set of methods, which allows a user to collect information presented on the World Wide Web (WWW). It is a similar technology which is used by search engines marked as Web Crawling. Although web scraping is not a new term, in years past the practice has been more commonly known as screen scraping, data mining, web harvesting, or similar variations.
In this project, web scraping is used to collect data from a website, which is listed as a '*free to scrape*' website. Other websites are not being scraped due to legal issues.

2. INTRODUCTION:

In theory, web scraping is the practice of gathering data through any means other than a program interacting with an API (or, obviously, through a human using a web browser). This is most commonly accomplished by writing an automated program that queries a web server, requests data (usually in the form of the HTML and other files that comprise web pages), and then parses that data to extract needed information.
In practice, web scraping encompasses a wide variety of programming techniques and technologies, such as data analysis and information security. Web scrapers are excellent at gathering and processing large amounts of data (among other things). Rather than viewing one page at a time through the narrow window of a monitor, you can view databases spanning thousands or even millions of pages at once. In addition, web scrapers can go places that traditional search engines cannot. A Google search for "cheapest flights to Boston" will result in a slew of advertisements and popular flight search sites. Google only knows what these websites say on their content pages, not the exact results of various queries entered into a flight search application. However, a well-developed web scraper can chart the cost of a flight to Boston over time, across a variety of websites, and tell you the best time to buy your ticket.

3. WEB SCRAPING:

Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web (WWW) and save it to a file system or database for later retrieval or analysis. Commonly, web data is scrapped utilizing Hypertext Transfer Protocol (HTTP) or through a web browser. This is

accomplished either manually by a user or automatically by a bot or web crawler. Due to the fact that an enormous amount of heterogeneous data is constantly generated on the WWW, web scraping is widely acknowledged as an efficient and powerful technique for collecting big data.

4. PURPOSE OF WEB SCRAPING:

4.1 Market analysis and research:

Data collection from online sources became one of the market research methods. It offers much faster response, compared to a classical surveying. Consumers are active in the online world and share their experience, frustration or motivation. Targeted data collection from e-shop and advertising servers helps to update Indexes. Which are based on frequently changed prices. With the increasing relevance and availability of on-line prices that we see today, it is natural to ask whether the prediction of the consumer price index (CPI), or related statistics, may usefully be computed more frequently than existing monthly schedules allow for.

4.2 Enterprise technologies:

Incompatible enterprise technologies are common by larger projects. Still a unified presentation of data from several systems is necessary. In some specific cases the solution is based on Web Scraping.

4.3 Opinion Poll:

Movie Producers collect data about their current blockbusters. Such data includes the user feedback, if this was shared on the movie portals in a review.

4.4 Human Resources Agencies:

Human resource (HR) departments in large companies process many jobs offers for their companies and try to match the position with prospective employees. It’s not sufficient to use only incoming vacancy requests from candidates. HR Departments also cooperate with 3rd Party companies, which can offer them own

directories of professionals. Contact mining is an important activity for such agencies.

4.5 Social Network mining:

Social media (such as blogs, online social networks, microblogs) has become one of the major data sources for quantitative communication research over the past decade. By employing simple programing tools, researchers can extract relevant messages from social media platforms for various research purposes.

4.6 Government Services:

The Monitoring of criminal activities on social websites and specific forums is an important source of information for Government Agencies and Law Enforcement bodies.

4.7 Corporate spying:

In the corporate context, web scraping allows for a company to review both their own and the appearance of competitors in headlines of news servers. A company can also collect details about competitors and even about its own employees.

4.8 Social Mining and Sentiment Analysis:

Social media is a new source of data that is significantly different from conventional ones. Social media data sets are mostly user-generated, and are big, interlinked, and heterogeneous.

5. METHODS OF WEB SCRAPING:

5.1 Manual Scraping:

Manual scraping is still an option in specific situations. These situations are:

- When the amount of data is minimal,

- When the data being scraped does not require a repetitive task,
- When setting up automated scraping would take longer than the data collection itself.
- Possibly security measures or specific characteristics of the website do not allow automated methods.

5.2 HTML Parsing:

Web sites don't always provide their data in comfortable formats such as .csv or .json files. HTML Pages are created by the server as a response to a user's request. At this point server software is not relevant, rather the output in the browser is important.

5.3 DOM Parsing:

Document Object Model (DOM) Parsing is an evolution of HTML Parsing based on developments of the language and browsers which lead to the introduction of the Document Object Model. DOM is heavily used for Cascading Stylesheets (CSS) and JavaScript. Integration of DOM revealed new possibilities for addressing some specific parts of the webpage.

5.4 XPath:

Similar addressing possibility as DOM provides XPath (XML Path Language). The name suggests a usage for XML documents. It is applicable also to HTML format. XPath requires a more precisely structured webpage than DOM and has the same possibility to address segments within the webpage.

5.5 APIs:

Application Programming Interface (API) expects an application as a communication partner. A standard HTTP Request sent to an API Endpoint returns an answer from server. Each API has its own specification and options. The format of the answer can be set as option in the request. The most widely used format for API communication is JSON.

6. TOOLS/PLATFORM, HARDWARE AND SOFTWARE REQUIREMENT SPECIFICATIONS:

6.1 Tools:

- Microsoft VS Code
- Python 3.7
- MS Office
- Internet Connection
- Beautiful Soup python module

6.2 Platform

- Windows 7/8/8.1/10

6.3 Hardware Requirements:

HDD	1 GB
Processor	Pentium 4 or newer processor
Memory	2 GB

6.4 Software Requirements:

Python	Version 3.X
Microsoft VS Code	Version 1.X
BeautifulSoup module	Version 4.X
Random module	Version 2.X
Requests module	Version 2.X

7. GOALS OF IMPLEMENTATION:

To extract data from website using concept and tools of web scraping in python programming language.

8. DESCRIPTION OF WEBSITE TO SCRAPE:

8.1 URL: http://quotes.toscrape.com/page/

8.2 Image of Website:

8.3 HTML Tags:

HTML	DESCRIPTION
	HTML div element, is a container for each quote.
	HTML span element. Contains the text/quote.
<small class="author" itemprop="author"></small>	HTML small element. Contains the Author name in text format.
	HTML anchor (a) element. Contains the Author specific page URL.
	HTML span element. Contains the Author's date of birth.

	HTML span element. Contains the Author's born location information.
<h3 class="author-title"> h3>	HTML h3 element. Contains the Author's name.
<li class="next">	HTML unordered list (ul-li) element. Contains the next pagination information.
	HTML anchor (a) element. Contains the pagination URL

9. PYTHON MODULES:

9.1 Beautiful Soup 4:

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with HTML/XML parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Three features make it powerful:

- Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application.

- Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. No encodings is required, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then encoding has to be specified.

- Beautiful Soup sits on top of popular Python parsers like 'lxml' and 'html5lib', allowing different parsing strategies or trade speed for flexibility.

9.2 Requests:

Requests allows to send organic, grass-fed HTTP/1.1 requests, without the need for manual labor. There's no need to manually add

query strings to URLs, or to form-encode of POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to urllib3.

9.3 Random:

This module implements pseudo-random number generators for various distributions.
For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

10. ALGORITHMS & FLOWCHART:

10.1 Algorithm:

- Driver Code

  - Import python3 modules → requests, random, beautifulsoup4
  - Get the url : http://quotes.toscrape.com/page/ and pass it onto "Module Scrape"
  - Loop:

    - Get the html using requests module get() method
    - Check status code (success = 200)
    - Parse the html fetched using beautiful soup 4 module and specify 'html.parser'
    - Check for pagination using beautiful soup method find() for html tag with class of "next"
    - If a page is found then scrape the page using scrapper() method of "Module Scrape"

  - Now get the data of all parsed html pages using get_data() method of "Module Scrape"
  - Loop:

- Initialize ‘Game’ class of “Module Game”
- Call game_logic() method of ‘Game’ class
- Continue until user wants to play again

➢ Module Scrape

- Import “Module Person”
- Declare class ‘Scrap’ with instance variables url and data[]
    - Declare scrapper() method as:
        - Method gets argument of ‘soup’ which the beautiful soup 4 object of a single page after parsing
        - Find all the html tag with class ‘quote’
        - For each successful find, initialize a Person using “Module Person”
        - Store the Person object in data[]
    - Declare get_data() method as:
        - Return the data[] information of ‘Scrap’ class

➢ Module Person

- Import python modules requests and beautiful soup 4
- Declare class ‘Person’ (with beautiful soup 4 html parsed soup object as argument) as:
    - Declare initialization function as:
        - Get the html tag with class ‘text’
        - Call get_text() method on soup object and store in instance variable ‘quote’
        - Get the html tag with class ‘author’
        - Call get_text() method on soup object and store in instance variable ‘name’
        - Get the anchor (a) html tag
        - Get the ‘href’ and store in instance variable ‘about_link’
    - Declare more_info() method as:

        - Scrape the about information of author using the ‘about_link’ instance variable
        - Scrape the birthdate and birth location of author and store the information

    - Declare get_quote() as:
        - Return the quote said by author

    - Declare get_name() as:
        - Return the name of author

    - Declare get_birthdate() as:
        - Return the birthdate of author

    - Declare get_location() as:
        - Return the birth location of author

    - Declare get_first_letter() as:
        - Return the first letter of author’s name

➢ Module Game

- Import python module → random
- Declare class ‘Game’ (with ‘Person’ class data) as:

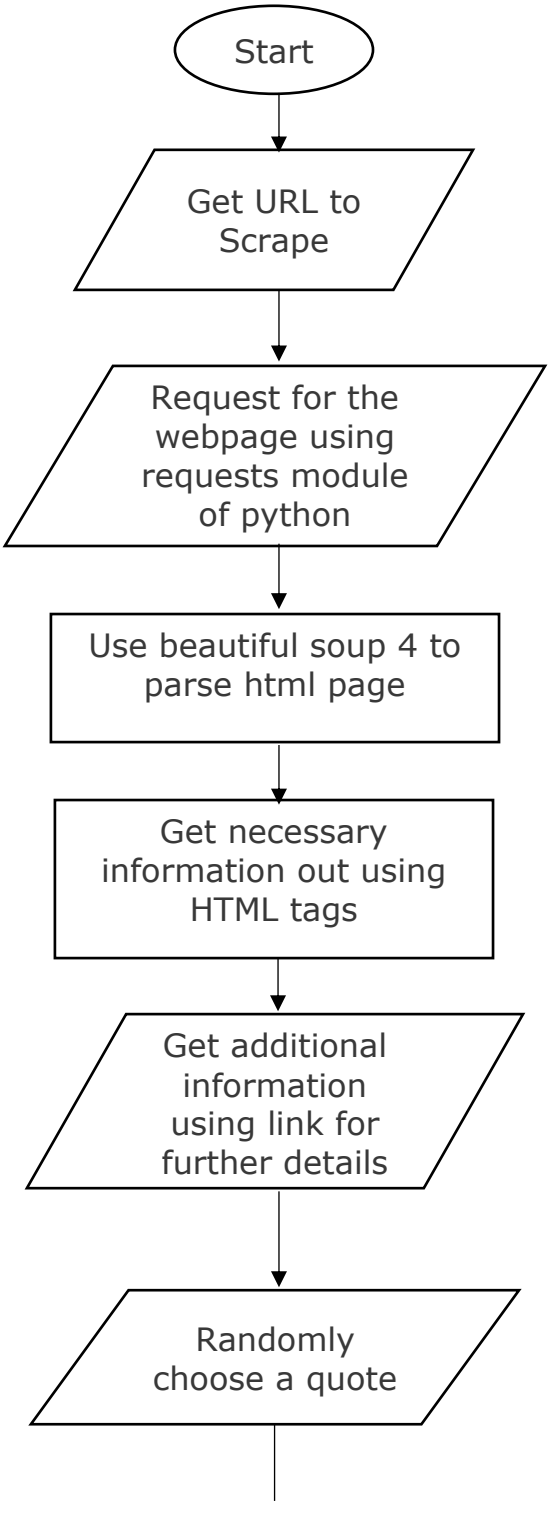
    - Initialize Game class with a random choice from data[]
    - Declare get_guess_count() method as:
        - Return the number of guess permitted

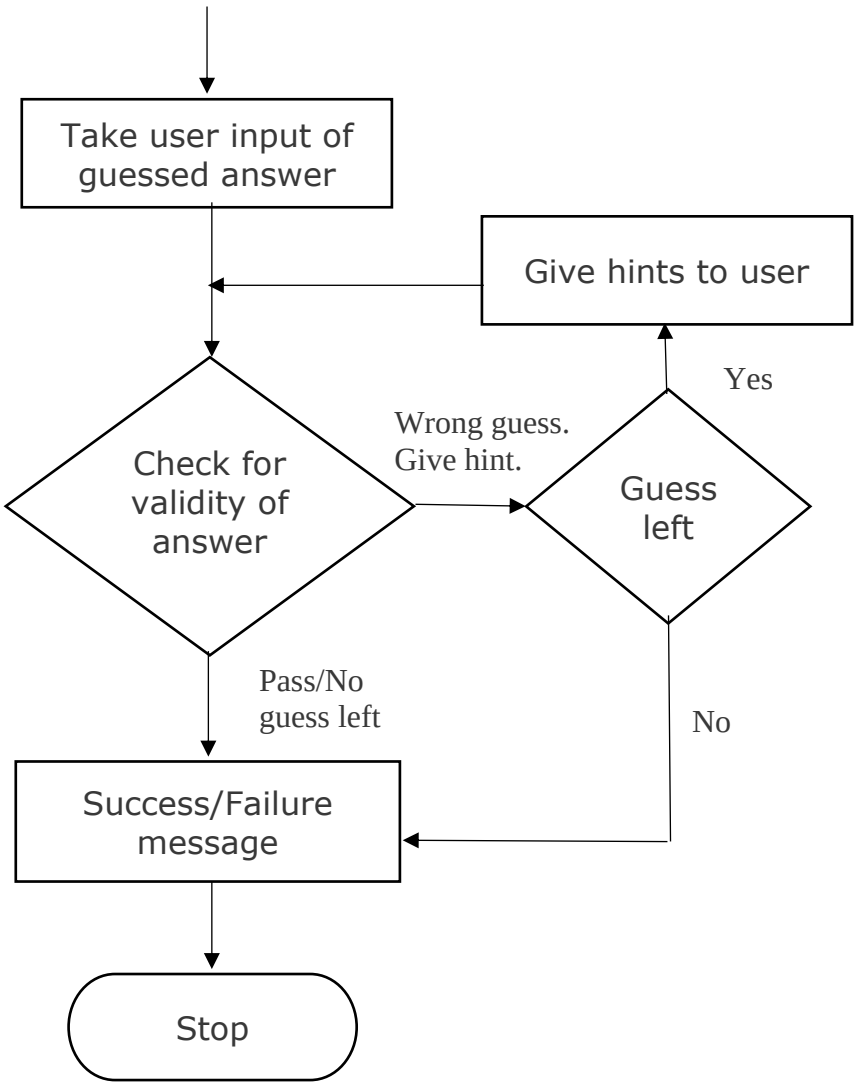
    - Declare decrease_guess_count() method as:
        - Decrease number of guesses left by one

    - Declare game_logic() method as:
        - Get a quote for playing game
        - Check if user has allowed guesses
        - If user fails in providing the correct answer, give hint by calling get_hint() method
        - Continue till no guess is left or user guessed correct answer

- Declare get_hint() method as:
    - Check number of guess left
    - For each failed guess give user some hint about correct answer such as birthdate information, birth location, starting letter of name

10.2 Flowchart





11. CONCLUSION:

Over the course of the project, knowledge about web scrapers has been gained, and using this knowledge, a number of things could be done differently if faced with this project again.

Although, at its core, the project did not deviate from the original plan, due to the spiders being specifically targeted to one site each, the choice of websites to target was highly limited. If faced with this problem again, an attempt to create a 'general purpose' spider would be made. This type of spider would need to intelligently find which HTML elements and classes contain the required information. This, however, is beyond the scope of this project.

With the backing of law enforcement agencies, spiders could be made to scrape website on an online domain and be allowed to follow links recursively.

12. FUTURE SCOPE:

With more time and added scope to the project, an intelligent system could be created to determine the context of the scraped messages. Limitation of dedicated site scraping can be improved by adding intelligent crawlers and thus more data.
But due to law and rules regarding web scraping and data mining, valid permissions are to be taken before performing such operations.

13. REFERENCES:

- Kevincurran Org
  https://kevincurran.org/dissertations/2018%20Thesis%20Evan%20Gallagher%20-%20Scraping%20Websites%20for%20Law.pdf

- Yanfei
  https://yanfei.site/docs/dpsa/references/PyWebScrapingBook.pdf

- Academia
  https://www.academia.edu/35901535/BACHELOR_PAPER_Web_Scraping_Data_Extraction_from_websites