Modern Education Society's
College of Engineering, Pune

NAME OF STUDENT	CLASS
SEMESTER/YEAR	ROLL NO
DATE OF PERFORMANCE	DATE OF SUBMISSION
EXAMINED BY	EXPERIMENT NO

Assignment No-4

Title: Write a program to implement simple web crawler.

Objectives: to implement simple web crawler.

Problem Statement: Write a program to implement simple web crawler.

Outcomes: Student can understand how to implement simple web crawler

Tools Required:

Hardware:

Software: Open source operating system

Theory:

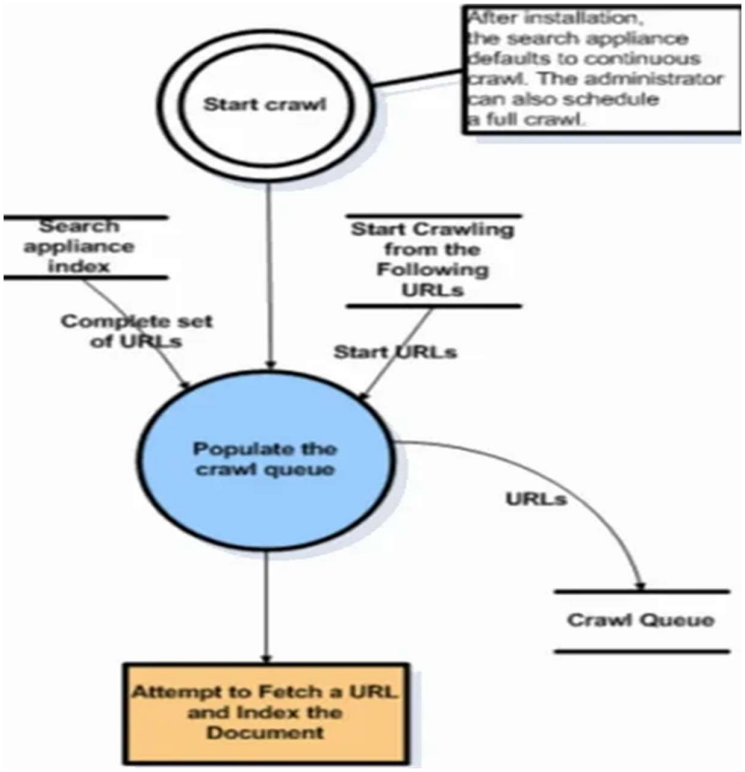
What Is A Web Crawler

A web crawler is an internet bot that indexes the content of websites (read the detailed definition on Wikipedia). It can automatically extract target information and data from websites and export data into structured formats (list/table/database).

You can view a web crawler as a particular program designed to crawl websites in orientation and glean data. However, you are unable to get the URL address of all web pages within a website containing many web pages in advance. Thus, what concerns us is how to fetch all the HTML web pages from a website.

Normally, we could define an entry page: One web page contains URLs of other web pages, and then we could retrieve these URLs from the current page and add

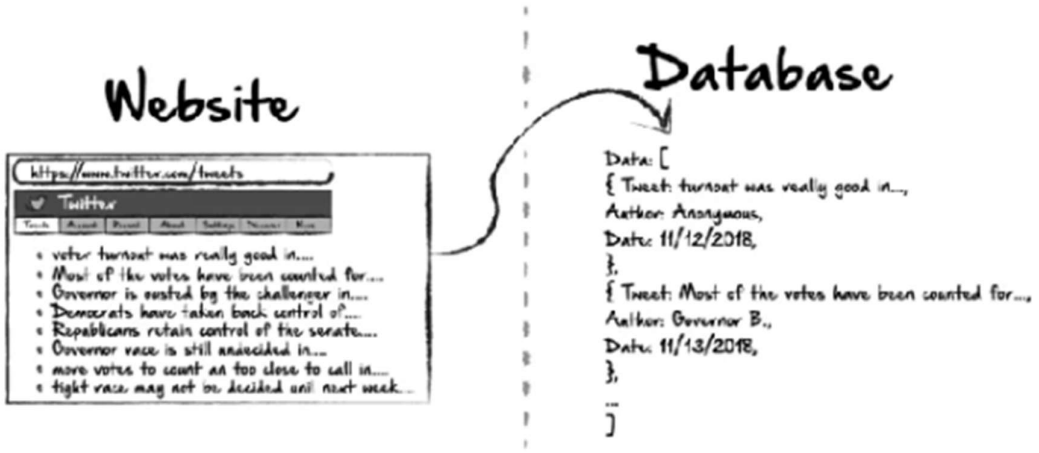
all of these affiliated URLs into the crawling queue. Next, we crawl another page and repeat the same process as the first one, recursively. Essentially, we could assume the crawling scheme is a depth-search or breadth-traversal. And as long as we could access the Internet and analyze the web page, we could crawl a website. Fortunately, most programming languages offer HTTP client libraries to crawl web pages, and we can even use regular expressions for HTML analysis.



Introduction:

Why Do You Need A Web Crawler

Imagine a world without Google Search. How long do you think it will take to get a recipe for chicken nuggets from the Internet? There are 2.5 quintillion bytes of data being created online each day. Without search engines like Google, it will be like looking for a needle in a haystack.



A search engine is a unique kind of web crawler that indexes websites and finds web pages for us. Besides search engines, you can also build a customized web crawler to help you achieve

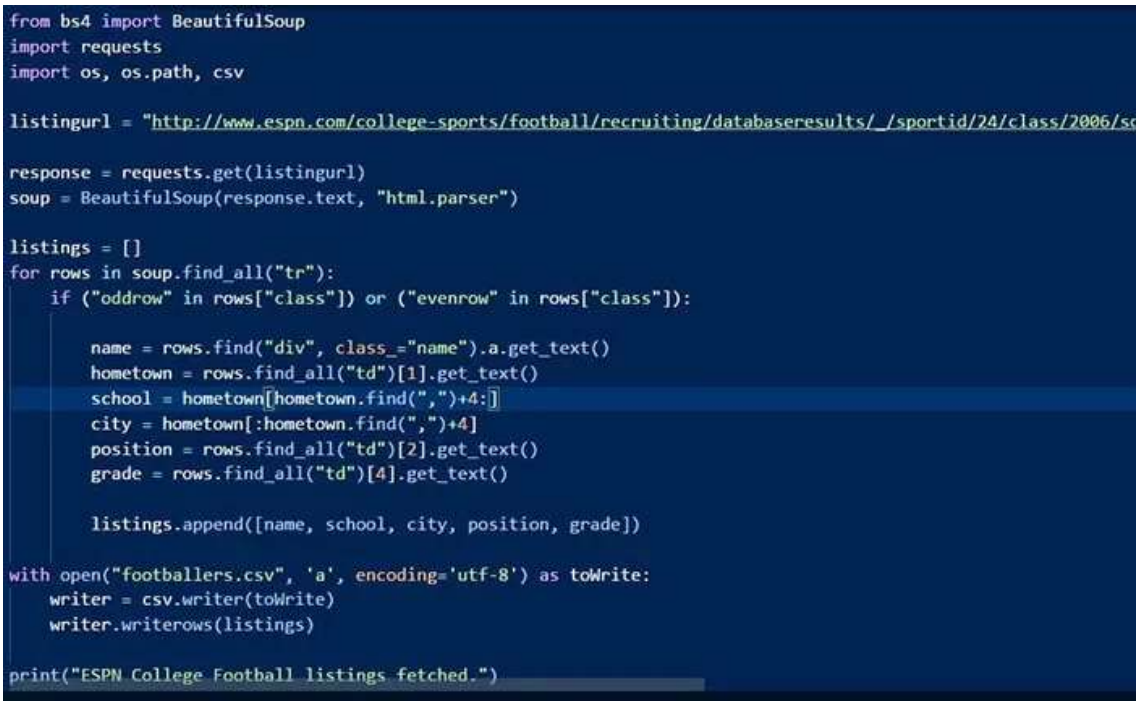
1. Content aggregation: It works to compile information on niche subjects from various resources into one single platform. As such, it is necessary to crawl popular websites to fuel your platform in time.

2. Sentiment analysis: It is also called opinion mining. As the name indicates, it is the process to analyze public attitudes toward one product or service. It requires a monotonic set of data to evaluate accurately. A web crawler can extract tweets, reviews, and comments for analysis.

3. Lead generation: Every business needs sales leads. That's how they survive and prosper. Let's say you plan to make a marketing campaign targeting a specific industry. You can scrape email, phone number, and public profiles from an exhibitor or attendee list of Trade Fairs, like attendees of the 2018 Legal Recruiting Summit.

Method 1: Build A Web Crawler with Coding Script

Writing scripts with computer languages is predominantly used by programmers. It can be as powerful as you create it to be. Here is an example of a snippet of bot code.



```
from bs4 import BeautifulSoup
import requests
import os, os.path, csv

listingurl = "http://www.espn.com/college-sports/football/recruiting/databaseresults/_/sportid/24/class/2006/sc

response = requests.get(listingurl)
soup = BeautifulSoup(response.text, "html.parser")

listings = []
for rows in soup.find_all("tr"):
    if ("oddrow" in rows["class"]) or ("evenrow" in rows["class"]):

        name = rows.find("div", class_="name").a.get_text()
        hometown = rows.find_all("td")[1].get_text()
        school = hometown[hometown.find(",")+4:]
        city = hometown[:hometown.find(",")+4]
        position = rows.find_all("td")[2].get_text()
        grade = rows.find_all("td")[4].get_text()

        listings.append([name, school, city, position, grade])

with open("footballers.csv", 'a', encoding='utf-8') as toWrite:
    writer = csv.writer(toWrite)
    writer.writerows(listings)

print("ESPN College Football listings fetched.")
```

3 steps to create a web crawler using Python

Step 1: Send an HTTP request to the URL of the webpage. It responds to your request by returning the content of web pages.

Step 2: Parse the webpage. A parser will create a tree structure of the HTML as the webpages are intertwined and nested together. A tree structure will help the bot follow the paths that we created and navigate through to get the information.

Step 3: Using the Python library to search the parse tree.

Among the computer languages for a web crawler, Python is an easy-to-implement compared to PHP and Java. It still has a steep learning curve that prevents many non-tech professionals from using it. Even though it is an economic solution to write your own, it's still not sustainable with regard to the extended learning cycle within a limited time frame.

Conclusion: Writing scripts can be painful as it has high initial and maintenance costs. No single web page is identical, and we need to write a script for every single site. It is not sustainable if you need to crawl many different websites. Besides, websites tend to change layouts and structures after some time. As a result, we have to debug and adjust the crawler accordingly. A free web crawler

like Octopuses or data crawler templates is more practical for beginners with less effort.

Conclusion: After completion of this experiment we have understand the concept and working of web crawler.

Questions:

1)What are the example of web crawler?