

<b>Name:-</b>	Pranav sathi, Virinchi Shettigar, Vignesh shinde, Jyoti bagul
<b>Batch</b>	C
<b>Exp no:-</b>	8

### **AIM:**

To demonstrate behavior of web crawlers and web spiders.

### **THEORY:**

#### **Web Crawler Bots:**

- Web crawlers, also known as spiders or search engine bots, are software programs that systematically browse the internet to download and index content from web pages.
- The primary purpose of web crawlers is to gather information about web content, which is later used to provide relevant search results when users query search engines like Google or Bing.
- Web crawlers are mainly operated by search engines and are responsible for organising and cataloguing the vast amount of information on the internet.
- They follow hyperlinks from one webpage to another, effectively "crawling" the web and discovering new content.

#### **Search Indexing:**

- Search indexing is the process of creating a structured database (index) of web content. It allows search engines to quickly retrieve relevant information in response to user search queries.
- Indexing involves analysing the text and metadata of web pages, including titles, descriptions, and keywords.
- Search engines add the words on web pages to their index, excluding common stop words such as "a," "an," and "the."

#### **How Web Crawlers Work:**

Web crawlers start from a seed, a list of known URLs, and then crawl the webpages at those URLs. They discover new URLs from the pages they visit and add them to their list for future crawling.

- Crawlers prioritise pages based on factors like the number of inbound links, visitor traffic, and other indicators of a page's importance and relevance.
- They revisit web pages periodically to ensure the index reflects the latest content updates.
- Web crawlers also respect the robots.txt protocol, a text file that specifies which pages bots can or cannot crawl on a website.

### **Web Crawler Access to Web Properties:**

- Decisions about whether to allow or disallow web crawlers to access web properties depend on the website owner's preferences and technical considerations.
- Web crawlers consume server resources and bandwidth, so website operators may limit indexing to avoid overloading their servers or incurring excessive costs.
- Some web pages are intentionally blocked from indexing by adding "no index" or "disallow" tags in the page or the robots.txt file.

### **Difference Between Web Crawling and Web Scraping:**

- Web crawling is the automated process of systematically browsing the internet to index web content for search engines.
- Web scraping involves extracting specific data from web pages without permission, often for purposes like data extraction or scraping content for malicious use.
- Web crawlers follow links and continuously discover new content, while web scrapers are more focused on specific pages or websites.
- Web crawlers respect server resource limitations and follow robots.txt rules, whereas web scrapers may not consider server load and may disregard rules, potentially overloading web servers.

## Crawling\_spiders.py

```
from scrapy.spiders import CrawlSpider, Rule from scrapy.linkextractors import LinkExtractor import json
from scrapy.crawler import CrawlerProcess

class CrawlingSpider(CrawlSpider): name = 'mycrawler'
allowed_domains = ['books.toscrape.com'] start_urls = ['http://books.toscrape.com/'] rules = (
    Rule(LinkExtractor(allow=r"catalogue/"), callback='parse_page'),
)

def parse_page(self, response): yield {
    'title': response.css('h1::text').get(), 'price':
    response.css('.price_color::text').get(), 'availability':
    response.css(".availability::text")[1].get().replace("\n", "").replace(" ", ""),
}

if name == " main ":
    process = CrawlerProcess(settings={ 'FEED_FORMAT': 'json',
    'FEED_URI': 'output.json', # Specify the file name for the JSON output
    })
    process.crawl(CrawlingSpider) process.start()
```

## Scrapy\_to\_sqlite.py

```
from scrapy.crawler import CrawlerProcess
from scrapy.spiders import CrawlSpider, Rule from scrapy.linkextractors import LinkExtractor import json
import sqlite3

class CrawlingSpider(CrawlSpider): name = 'mycrawler'
allowed_domains = ['books.toscrape.com'] start_urls = ['http://books.toscrape.com/'] rules = (
    Rule(LinkExtractor(allow=r"catalogue/"), callback='parse_page'),
)

def parse_page(self, response): yield {
    'title': response.css('h1::text').get(), 'price':
    response.css('.price_color::text').get(), 'availability':
    response.css(".availability::text")[1].get().replace("\n", "").replace(" ", ""),
}

if name == " main ":
    process = CrawlerProcess(settings={ 'FEED_URI': 'output.json', # Specify the file name for the JSON output
    })
    process.crawl(CrawlingSpider) process.start()

# Read data from the JSON file
with open('output.json', 'r') as file: data = json.load(file)

# Create SQLite connection and cursor
conn = sqlite3.connect('book_data.db') # Replace 'book_data.db' with your desired database name
cursor = conn.cursor()

# Create a table to store the data cursor.execute('''CREATE TABLE IF NOT EXISTS
BookData (
id INTEGER PRIMARY KEY
AUTOINCREMENT,
title TEXT, price TEXT,
availability TEXT
)''')

# Insert data into the database for item in data:
cursor.execute('''INSERT INTO BookData (title, price, availability)
VALUES (?, ?, ?)''',
(item['title'], item['price'], item['availability']))

# Commit changes and close connection conn.commit()
conn.close()
```

## Sqlite\_data\_insert.py

```
import json
```

```
import sqlite3

def insert_data_to_sqlite():
    # Read data from the JSON file
    with open('output.json', 'r') as file: data = json.load(file)

    # Create SQLite connection and cursor
    conn = sqlite3.connect('book_data.db') # Replace 'book_data.db' with your desired database name
    cursor = conn.cursor()

    # Create a table to store the data if it doesn't exist
    cursor.execute('''CREATE TABLE IF NOT EXISTS BookData (

AUTOINCREMENT,

id INTEGER PRIMARY KEY

title TEXT, price TEXT,
availability TEXT
)''')

    # Insert data into the database for item in data:
    cursor.execute('''INSERT INTO BookData (title, price, availability)
VALUES (?, ?, ?)''',
(item['title'], item['price'], item['availability']))

    # Commit changes and close connection conn.commit()
    conn.close()

if name == " main ": insert_data_to_sqlite()
```

## Query\_database.py

```
import sqlite3

def query_database():
    conn = sqlite3.connect('book_data.db') # Replace 'book_data.db' with your database file name
    cursor = conn.cursor()

    # Execute SQL queries cursor.execute("SELECT * FROM BookData;") rows = cursor.fetchall()

    # Process or print retrieved data for row in rows:
    print(row)

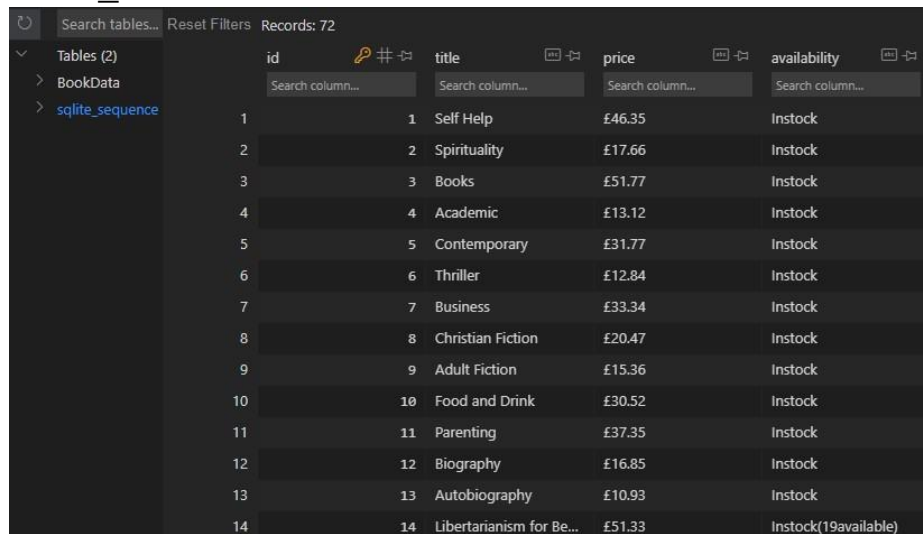
    # Close connection conn.close()

    # Call the function to perform database operations if name == " main ":
    query_database()
```

## Output.json

```
1 [{"title": "Self Help", "price": "\u00a346.35", "availability": "Instock"},
2 {"title": "Spirituality", "price": "\u00a317.66", "availability": "Instock"},
3 {"title": "Books", "price": "\u00a351.77", "availability": "Instock"},
4 {"title": "Academic", "price": "\u00a313.12", "availability": "Instock"},
5 {"title": "Contemporary", "price": "\u00a331.77", "availability": "Instock"},
6 {"title": "Thriller", "price": "\u00a312.84", "availability": "Instock"},
7 {"title": "Business", "price": "\u00a333.34", "availability": "Instock"},
8 {"title": "Christian Fiction", "price": "\u00a320.47", "availability": "Instock"},
9 {"title": "Adult Fiction", "price": "\u00a315.36", "availability": "Instock"},
10 {"title": "Food and Drink", "price": "\u00a330.52", "availability": "Instock"},
11 {"title": "Parenting", "price": "\u00a337.35", "availability": "Instock"},
12 {"title": "Biography", "price": "\u00a316.85", "availability": "Instock"},
13 {"title": "Autobiography", "price": "\u00a310.93", "availability": "Instock"},
14 {"title": "Libertarianism for Beginners", "price": "\u00a351.33", "availability": "Instock(19available)"},
15 {"title": "It's Only the Himalayas", "price": "\u00a345.17", "availability": "Instock(19available)"},
16 {"title": "Olio", "price": "\u00a323.88", "availability": "Instock(19available)"},
17 {"title": "Humor", "price": "\u00a344.07", "availability": "Instock"},
18 {"title": "Mesaerion: The Best Science Fiction Stories 1800-1849", "price": "\u00a337.59", "availability": "Instock(19"},
19 {"title": "Our Band Could Be Your Life: Scenes from the American Indie Underground, 1981-1991", "price": "\u00a357.25"},
20 {"title": "Rip it Up and Start Again", "price": "\u00a335.02", "availability": "Instock(19available)"},
21 {"title": "All products", "price": "\u00a312.84", "availability": "Instock"},
22 {"title": "Scott Pilgrim's Precious Little Life (Scott Pilgrim #1)", "price": "\u00a352.29", "availability": "Instock"},
23 {"title": "Set Me Free", "price": "\u00a317.46", "availability": "Instock(19available)"},
24
```

## Book\_data.db



	id	title	price	availability
1	1	Self Help	£46.35	Instock
2	2	Spirituality	£17.66	Instock
3	3	Books	£51.77	Instock
4	4	Academic	£13.12	Instock
5	5	Contemporary	£31.77	Instock
6	6	Thriller	£12.84	Instock
7	7	Business	£33.34	Instock
8	8	Christian Fiction	£20.47	Instock
9	9	Adult Fiction	£15.36	Instock
10	10	Food and Drink	£30.52	Instock
11	11	Parenting	£37.35	Instock
12	12	Biography	£16.85	Instock
13	13	Autobiography	£10.93	Instock
14	14	Libertarianism for Be...	£51.33	Instock(19available)

### CONCLUSION:-

By performing this experiment, I understood the concept of web crawlers. I understood what they are and how they differ from web scrapers. I understood the two ways to crawl that is xpath and csspath and implemented one of them.