L6 Data Collection

In this practical, you learn

- how to scrap web pages using Scrapy and extract the content using XPath, regex, and CSSSelector with LXML

Useful Tools for Regex, XPath and CSSSelector development Crawling and extraction rely heavily on the usage of XPath, and CSS Selector. However developing these patterns from scratch might be challenging, you might find some of the following tools useful.

XPath Wizard https://chrome.google.com/webstore/detail/xpath-helper-wizard/jadhpggafkbmpdpmpgigopmodldgfcki?hl=en

Selector Gadget https://chrome.google.com/webstore/detail/selectorgadget/mhjhnkcfbdhnjickkkdbjoemdmbfginb?hl=en

Focused crawl

First let's consider a simple crawler which crawl a quotes website using focused crawl strategy. quotes.toscrape.com

Suppose that by navigating the website, we are able to guess the list of pages is in the shape of `http://quotes.toscrape.com/page/1/`, `http://quotes.toscrape.com/page/2/`,

The quotes.toscrape example is inspired by [https://www.jitsejan.com/using-scrapy-in-jupyter-notebook.html].

Recall from the lecture note that a focused crawl behaves as follows,

1. For each URL `u` in the list of seed URLs,
    1. extract the needed content from `u`.

We can use a function or just hard coding to generate the sequence of start / seed URLs.

Note that a focused crawl does not follow links in the pages. We get a page from the list, and extract the needed content.

First of all we need to define some writer classes, which help to debug or save the output of the extract.

- `ConsoleWriterPipeline` receives the extract result from the spider and prints out the content.
- `JsonWriterPipeline` receives the extract result from the spider and appends them into a JSON Line file, (each line is a json)

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
data_dir_path='/content/drive/My Drive/Data/DS6/'
```

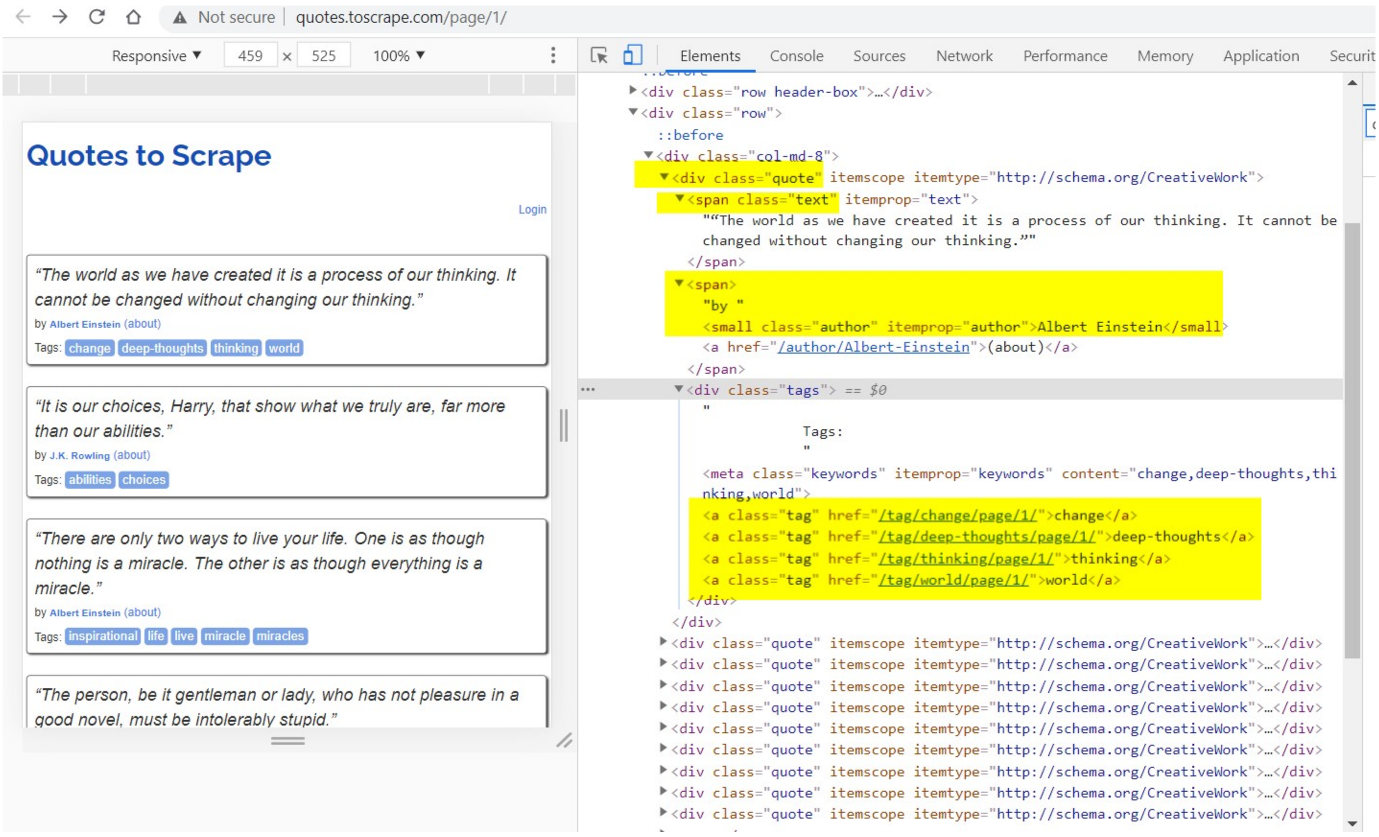
```
import lxml.etree

import json

# receives the extract result from the spider and prints out the content
class ConsoleWriterPipeline(object):
    def open_spider(self, spider):
        None
    def close_spdier(self, spider):
        None

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        print(line)
        return item

# receives the extract result from the spider and appends them into a JSON Line fil
class JsonWriterPipeline(object):
    def open_spider(self, spider):
        self.file = open(data_dir_path+'result.json', 'w')

    def close_spider(self, spider):
        print('JSON File Generated')
        self.file.close()

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line)
        return item
```

Next we define our spider, `QuoteSpider` is a focused spider.

It reads a list of URLs and calls `parse()` for each page (`response`) given by the link.

Note that for each link, we find multiple quotes. Hence, in `parse()` we use a CSS selector to retrieve the list of all `div` elements that containing the quotes, one quote per element.

The `yield` statement constructs the result JSON object that will be consumed by the downstream writer, in this case we use `ConsoleWriterPipeline`.


Not secure | quotes.toscrape.com/page/1/
Responsive ▾ 459 × 525 100% ▾
Elements Console Sources Network Performance Memory Application Securit

Quotes to Scrape

Login

“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”
by Albert Einstein (about)
Tags: change deep-thoughts thinking world

“It is our choices, Harry, that show what we truly are, far more than our abilities.”
by J.K. Rowling (about)
Tags: abilities choices

“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”
by Albert Einstein (about)
Tags: inspirational life live miracle miracles

“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”

```
▶<div class="row header-box">…</div>
▼<div class="row">
   ::before
 ▼<div class="col-md-8">
   ▼<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">
     ▼<span class="text" itemprop="text">
        "“The world as we have created it is a process of our thinking. It cannot be
        changed without changing our thinking.”"
      </span>
     ▼<span>
        "by "
        <small class="author" itemprop="author">Albert Einstein</small>
        <a href="/author/Albert-Einstein">(about)</a>
      </span>
…    ▼<div class="tags"> == $0
        "
                Tags:
                "
        <meta class="keywords" itemprop="keywords" content="change,deep-thoughts,thi
        nking,world">
        <a class="tag" href="/tag/change/page/1/">change</a>
        <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
        <a class="tag" href="/tag/thinking/page/1/">thinking</a>
        <a class="tag" href="/tag/world/page/1/">world</a>
      </div>
    </div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
   ▶<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
```

```
for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').get(),
                'author': quote.css('span small::text').get(),
                'tags': quote.css('div.tags a.tag::text').getall(),
            }
```

```
!pip install scrapy
```

```
    Collecting itemloaders>=1.0.1
      Downloading itemloaders-1.0.4-py3-none-any.whl (11 kB)
    Collecting service-identity>=16.0.0
      Downloading service_identity-21.1.0-py2.py3-none-any.whl (12 kB)

    Collecting parsel>=1.5.0
      Downloading parsel-1.6.0-py2.py3-none-any.whl (13 kB)
    Collecting zope.interface>=4.1.3
      Downloading zope.interface-5.4.0-cp37-cp37m-manylinux2010_x86_64.whl (251 kE
         |████████████████████████████████| 251 kB 41.2 MB/s
    Collecting protego>=0.1.15
      Downloading Protego-0.2.1-py2.py3-none-any.whl (8.2 kB)
    Collecting Twisted>=17.9.0
      Downloading Twisted-22.4.0-py3-none-any.whl (3.1 MB)
         |████████████████████████████████| 3.1 MB 33.2 MB/s
    Collecting tldextract
      Downloading tldextract-3.3.0-py3-none-any.whl (93 kB)
         |████████████████████████████████| 93 kB 2.1 MB/s
    Collecting pyOpenSSL>=16.2.0
      Downloading pyOpenSSL-22.0.0-py2.py3-none-any.whl (55 kB)
         |████████████████████████████████| 55 kB 3.9 MB/s
```

```
Collecting cssselect>=0.9.1
  Downloading cssselect-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting PyDispatcher>=2.0.5
  Downloading PyDispatcher-2.0.5.zip (47 kB)
     |████████████████████████████████| 47 kB 4.3 MB/s
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-pack
Collecting jmespath>=0.9.5
  Downloading jmespath-1.0.0-py3-none-any.whl (23 kB)
Requirement already satisfied: six>=1.6.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyasn1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyasn1-modules in /usr/local/lib/python3.7/dist
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: typing-extensions>=3.6.5 in /usr/local/lib/pyth
Collecting hyperlink>=17.1.1
  Downloading hyperlink-21.0.0-py2.py3-none-any.whl (74 kB)
     |████████████████████████████████| 74 kB 3.4 MB/s
Collecting incremental>=21.3.0
  Downloading incremental-21.3.0-py2.py3-none-any.whl (15 kB)
Collecting Automat>=0.8.0
  Downloading Automat-20.2.0-py2.py3-none-any.whl (31 kB)
Collecting constantly>=15.1
  Downloading constantly-15.1.0-py2.py3-none-any.whl (7.9 kB)
Requirement already satisfied: idna>=2.5 in /usr/local/lib/python3.7/dist-pack
Collecting requests-file>=1.4
  Downloading requests_file-1.5.1-py2.py3-none-any.whl (3.7 kB)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dis
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Building wheels for collected packages: PyDispatcher
  Building wheel for PyDispatcher (setup.py) ... done
  Created wheel for PyDispatcher: filename=PyDispatcher-2.0.5-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/2d/18/21/3c6a732eaa69a339198e08
Successfully built PyDispatcher
Installing collected packages: w3lib, cssselect, zope.interface, requests-file
Successfully installed Automat-20.2.0 PyDispatcher-2.0.5 Twisted-22.4.0 consta
```

```python
import logging
import scrapy
from scrapy.crawler import CrawlerProcess

class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
        'http://quotes.toscrape.com/page/2/',
    ]
    custom_settings = {
        'LOG_LEVEL': logging.WARNING,                            # Default : Debug
        'ITEM_PIPELINES': {'__main__.JsonWriterPipeline': 1}  # Used for pipeline
    }

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').get(),
                'author': quote.css('span small::text').get(),
                'tags': quote.css('div.tags a.tag::text').getall(),
            }
```

In the following, we create a process which will start the crawler. By uncommenting and running the below code, we perform the focused crawl the web site. The result will be printed in the output sessoin. In case it does not stop. You consider click the "Block Square" button below the menu bar to stop the kernel.

Note In case you hit the `ReactorNotRestartable:` error, you should comment away another crawler processes in this note book and restart the kernel.

User Agent is the runner that we use to execute the crawling process.

For more details, refer to https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

```
# uncomment me and run
# '''
quotes_crawler_process = CrawlerProcess({
    'USER_AGENT': 'Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1)'
})

quotes_crawler_process.crawl(QuotesSpider)
quotes_crawler_process.start()
# '''
```

```
    2022-05-25 12:13:08 [scrapy.utils.log] INFO: Scrapy 2.6.1 started (bot: scrapy
    2022-05-25 12:13:08 [scrapy.utils.log] INFO: Versions: lxml 4.2.6.0, libxml2 2
    2022-05-25 12:13:08 [scrapy.crawler] INFO: Overridden settings:
    {'LOG_LEVEL': 30,
     'USER_AGENT': 'Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1)'}
    JSON File Generated
```

Exercise

When you are happy with result, you may modify the `QuotesSpider` class to use `JSONWriterPipeline` to save the result in a file.

Question

Manually getting the list of input URLs for a focused crawl could be challenging? Is there anyway to automate it?

In the following example, we are going to BBC and get the headline and introduction from all the

General Crawl - News Crawler

Restart Runtime to avoid ReactorNotRestartable error.

At the http://www.bbc.co.uk/news/technology/ page parse the articles.

Get each article text for headline and introduction.

Setup the parsing result either console or json file.

```
import lxml.etree

import json

class ConsoleWriterPipeline(object):
    def open_spider(self, spider):
        None
    def close_spider(self, spider):
        None

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        print(line)
        return item

class JsonWriterPipeline(object):
    def open_spider(self, spider):
        self.file = open(data_dir_path+'newsresult.json', 'w')

    def close_spider(self, spider):
        print('JSON File Generated')
        self.file.close()

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line)
        return item
```

Define the start Url

```
"http://www.bbc.co.uk/news/technology/"
```

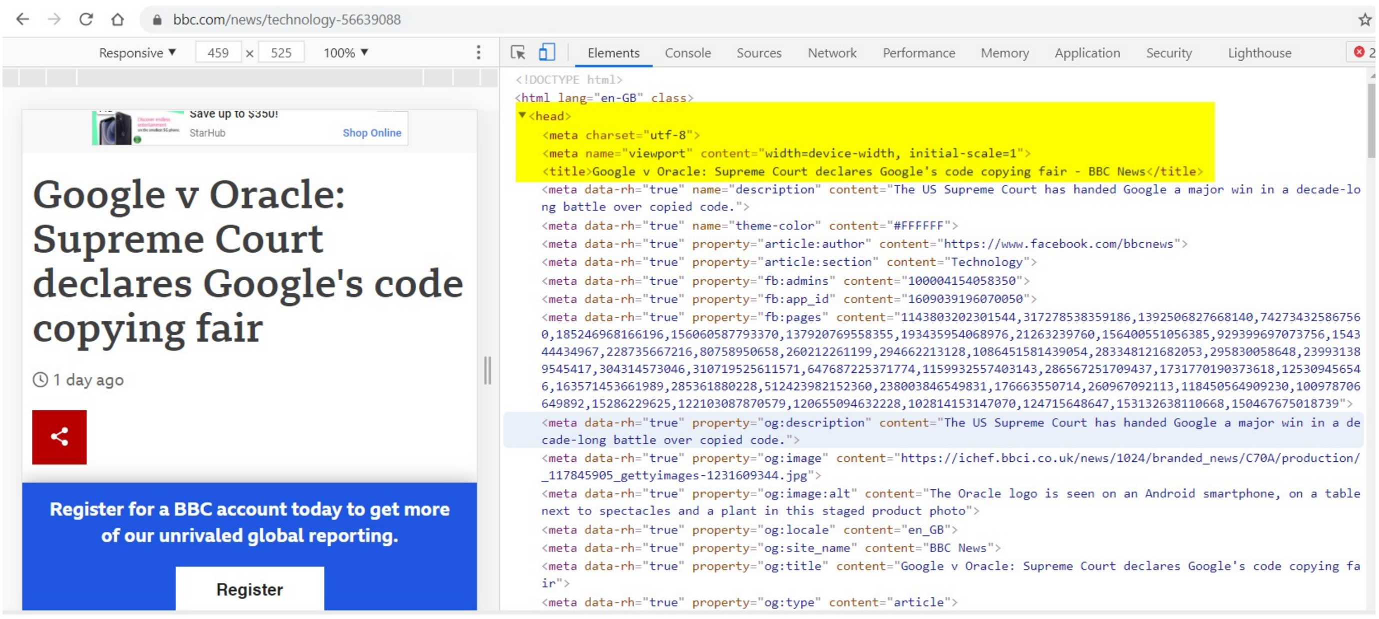
Set the rule for the parsing in the URL

```
Rule(LinkExtractor(allow=['/technology-\d+'])
```

Parsing function to extract each article headline and introduction

```
        story = NewsItem()
        story['headline'] = response.xpath('//head/title/text()').get()
        story['intro'] = response.xpath('//p/b/text()').get()

        yield {
            "headline":story['headline'],
            "intro":story['intro']
            }
```

Tag for the Headline



bbc.com/news/technology-56639088

Responsive 459 × 525 100%

Google v Oracle: Supreme Court declares Google's code copying fair

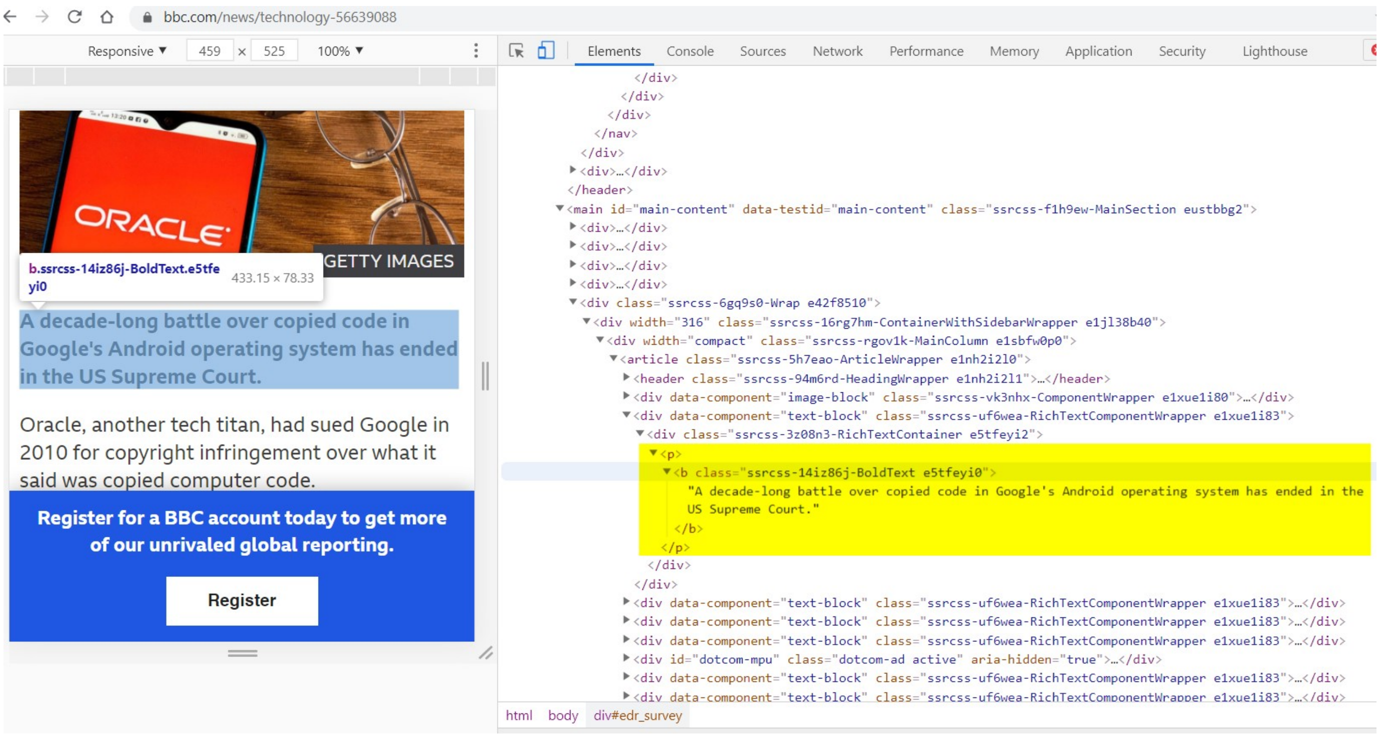
1 day ago

Register for a BBC account today to get more of our unrivaled global reporting.

Register

```
<!DOCTYPE html>
<html lang="en-GB" class>
▼<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title>Google v Oracle: Supreme Court declares Google's code copying fair - BBC News</title>
   <meta data-rh="true" name="description" content="The US Supreme Court has handed Google a major win in a decade-lo
   ng battle over copied code.">
   <meta data-rh="true" name="theme-color" content="#FFFFFF">
   <meta data-rh="true" property="article:author" content="https://www.facebook.com/bbcnews">
   <meta data-rh="true" property="article:section" content="Technology">
   <meta data-rh="true" property="fb:admins" content="100004154058350">
   <meta data-rh="true" property="fb:app_id" content="1609039196070050">
   <meta data-rh="true" property="fb:pages" content="1143803202301544,317278538359186,1392506827668140,74273432586756
   0,185246968166196,156060587793370,137920769558355,193435954068976,21263239760,156400551056385,929399697073756,1543
   44434967,228735667216,80758950658,260212261199,294662213128,1086451581439054,283348121682053,295830058648,23993138
   9545417,304314573046,310719525611571,647687225371774,1159932557403143,286567251709437,1731770190373618,12530945654
   6,163571453661989,285361880228,512423982152360,238003846549831,176663550714,260967092113,118450564909230,100978706
   649892,15286229625,122103087870579,120655094632228,102814153147070,124715648647,153132638110668,150467675018739">
   <meta data-rh="true" property="og:description" content="The US Supreme Court has handed Google a major win in a de
   cade-long battle over copied code.">
   <meta data-rh="true" property="og:image" content="https://ichef.bbci.co.uk/news/1024/branded_news/C70A/production/
   _117845905_gettyimages-1231609344.jpg">
   <meta data-rh="true" property="og:image:alt" content="The Oracle logo is seen on an Android smartphone, on a table
   next to spectacles and a plant in this staged product photo">
   <meta data-rh="true" property="og:locale" content="en_GB">
   <meta data-rh="true" property="og:site_name" content="BBC News">
   <meta data-rh="true" property="og:title" content="Google v Oracle: Supreme Court declares Google's code copying fa
   ir">
   <meta data-rh="true" property="og:type" content="article">
```

Tag for the introduction



bbc.com/news/technology-56639088

Responsive 459 × 525 100%

GETTY IMAGES

b.ssrcss-14iz86j-BoldText.e5tfeyi0 433.15 × 78.33

A decade-long battle over copied code in Google's Android operating system has ended in the US Supreme Court.

Oracle, another tech titan, had sued Google in 2010 for copyright infringement over what it said was copied computer code.

Register for a BBC account today to get more of our unrivaled global reporting.

Register

```
                </div>
              </div>
            </div>
          </nav>
        </div>
      ▶<div>…</div>
      </header>
    ▼<main id="main-content" data-testid="main-content" class="ssrcss-f1h9ew-MainSection eustbbg2">
      ▶<div>…</div>
      ▶<div>…</div>
      ▶<div>…</div>
      ▶<div>…</div>
      ▼<div class="ssrcss-6gq9s0-Wrap e42f8510">
        ▼<div width="316" class="ssrcss-16rg7hm-ContainerWithSidebarWrapper e1jl38b40">
          ▼<div width="compact" class="ssrcss-rgov1k-MainColumn e1sbfw0p0">
            ▼<article class="ssrcss-5h7eao-ArticleWrapper e1nh2i210">
              ▶<header class="ssrcss-94m6rd-HeadingWrapper e1nh2i211">…</header>
              ▶<div data-component="image-block" class="ssrcss-vk3nhx-ComponentWrapper e1xue1i80">…</div>
              ▼<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">
                ▼<div class="ssrcss-3z08n3-RichTextContainer e5tfeyi2">
                  ▼<p>
                    ▼<b class="ssrcss-14iz86j-BoldText e5tfeyi0">
                        "A decade-long battle over copied code in Google's Android operating system has ended in the
                        US Supreme Court."
                      </b>
                    </p>
                  </div>
                </div>
              ▶<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">…</div>
              ▶<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">…</div>
              ▶<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">…</div>
              ▶<div id="dotcom-mpu" class="dotcom-ad active" aria-hidden="true">…</div>
              ▶<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">…</div>
              ▶<div data-component="text-block" class="ssrcss-uf6wea-RichTextComponentWrapper e1xue1i83">…</div>
html body div#edr_survey
```

```
import logging
import scrapy
from scrapy.spiders import Rule, CrawlSpider
from scrapy.linkextractors import LinkExtractor

class NewsItem(scrapy.Item):
  # define the fields for your item here like:
  headline = scrapy.Field()
  intro = scrapy.Field()
  # url = scrapy.Field()

class NewsSpider(CrawlSpider):
  name = "bbcnews"
  allowed_domains = ["bbc.co.uk"]
  start_urls = ["http://www.bbc.co.uk/news/technology/",]
  custom_settings = {
      'LOG_LEVEL': logging.WARNING,
      'ITEM_PIPELINES': {'__main__.ConsoleWriterPipeline': 1} # Used for pipeline 1
      }
  rules = [Rule(LinkExtractor(allow=['/technology-\d+']), 'parse_story')]

  def parse_story(self, response):
    story = NewsItem()
    story['headline'] = response.xpath('//head/title/text()').get()
    story['intro'] = response.xpath('//p/b/text()').get()

    yield {
        "headline":story['headline'],
        "intro":story['intro']
        }
```

```
from scrapy.crawler import CrawlerProcess

hgw_crawler_process = CrawlerProcess({
    'USER_AGENT': 'Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1)'
})

hgw_crawler_process.crawl(NewsSpider)
hgw_crawler_process.start()
```

```
2022-05-25 12:32:49 [scrapy.utils.log] INFO: Scrapy 2.6.1 started (bot: scrapy
2022-05-25 12:32:49 [scrapy.utils.log] INFO: Versions: lxml 4.2.6.0, libxml2 2
2022-05-25 12:32:49 [scrapy.crawler] INFO: Overridden settings:
{'LOG_LEVEL': 30,
 'USER_AGENT': 'Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1)'}
{"headline": "Cryptocrash: \u2018I was arrested for knocking on Luna boss's do

{"headline": "Subsea internet cables could help detect earthquakes - BBC News"

{"headline": "Formula milk: Online groups hunt for baby milk during US shortag

{"headline": "Clearview AI fined in UK for illegally storing facial images - B
```

```
{"headline": "President Rodrigo Chaves says Costa Rica is at war with Conti ha

{"headline": "Nano ink solar cells allow tech to charge in any light - BBC New

{"headline": "BBC announces first ever Gaming Prom - BBC News", "intro": null}

{"headline": "'I've got bass in a backpack': VR revives 1989 rave culture - BE
```

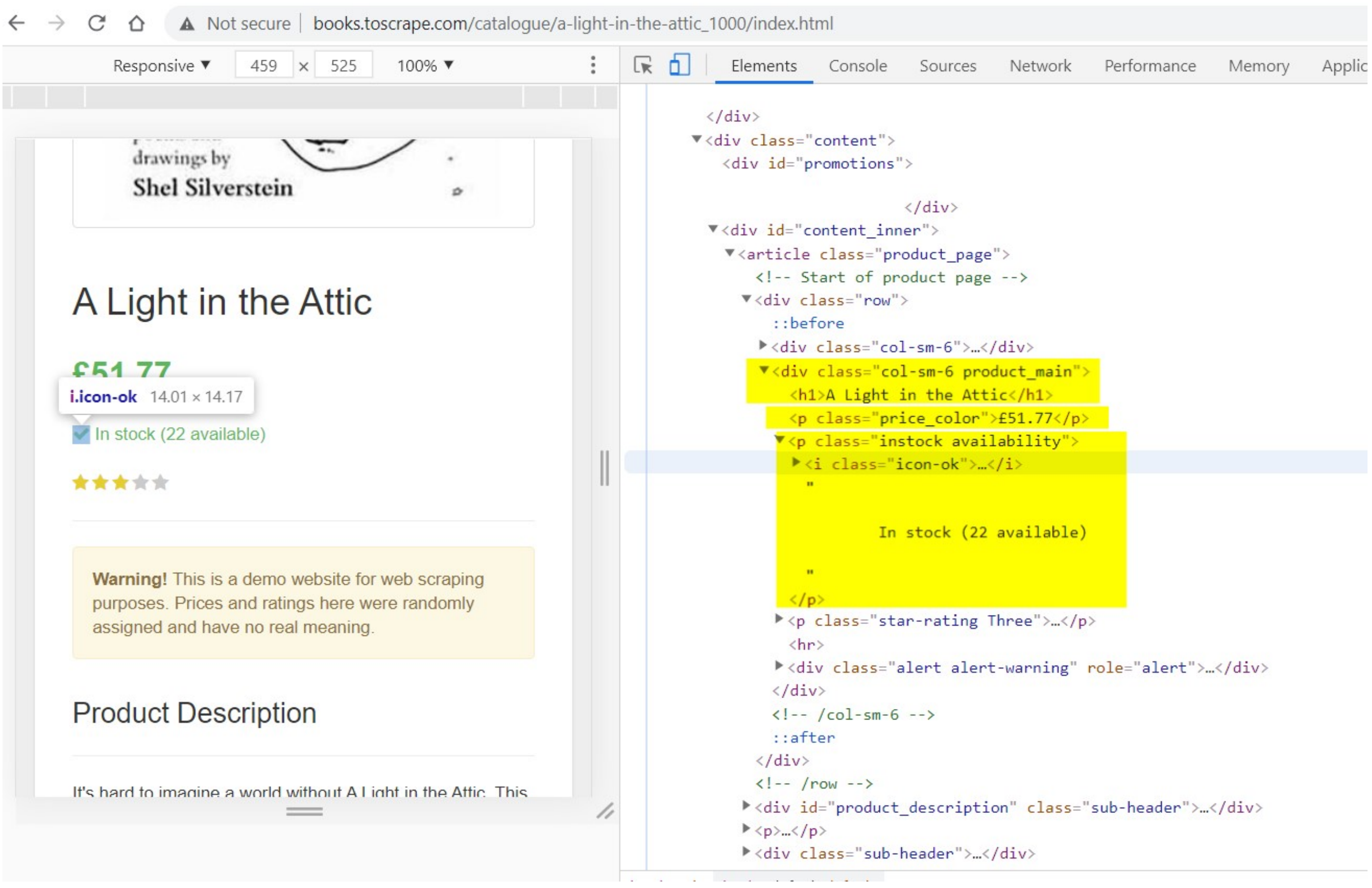
General Crawl - Book Crawler

A general crawler may have only one start URL, and typically two rules. It starts by added the start URL to its URL queue. it repeats the following until the URL queue is empty.

1. get a URL from the from the URL queue,
    1. rule 1. when a target URL is loaded, extract it.
    2. rule 2. when a non-target URL is loaded and add all (new) links in the page the URL queue.
2. remove the URL from the URL queue.

Goto http://books.toscrape.com.

Extract each book title, price and stock


Not secure | books.toscrape.com/catalogue/a-light-in-the-attic_1000/index.html
Responsive ▼ 459 × 525 100% ▼
Elements Console Sources Network Performance Memory Applic
drawings by Shel Silverstein
A Light in the Attic
£51.77
i.icon-ok 14.01 × 14.17
In stock (22 available)
Warning! This is a demo website for web scraping purposes. Prices and ratings here were randomly assigned and have no real meaning.
Product Description
It's hard to imagine a world without A Light in the Attic. This

```
    </div>
  ▼<div class="content">
      <div id="promotions">

          </div>
    ▼<div id="content_inner">
      ▼<article class="product_page">
          <!-- Start of product page -->
        ▼<div class="row">
            ::before
          ▶<div class="col-sm-6">…</div>
          ▼<div class="col-sm-6 product_main">
              <h1>A Light in the Attic</h1>
              <p class="price_color">£51.77</p>
            ▼<p class="instock availability">
              ▶<i class="icon-ok">…</i>
                "

                     In stock (22 available)

                "
              </p>
            ▶<p class="star-rating Three">…</p>
              <hr>
            ▶<div class="alert alert-warning" role="alert">…</div>
            </div>
            <!-- /col-sm-6 -->
            ::after
          </div>
          <!-- /row -->
        ▶<div id="product_description" class="sub-header">…</div>
        ▶<p>…</p>
        ▶<div class="sub-header">…</div>
```

```
yield {
            'title': response.css('.product_main h1::text').get(),
            'price': response.css('.product_main p.price_color::text').re_first('£(.*)'),
            'stock': int(
                ''.join(
                    response.css('.product_main .instock.availability ::text').re('(\d+)')
                )
            ),
        }



import lxml.etree

import json

class ConsoleWriterPipeline(object):
    def open_spider(self, spider):
        None
    def close_spdier(self, spider):
        None

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        print(line)
        return item

class JsonWriterPipeline(object):
    def open_spider(self, spider):
        self.file = open(data_dir_path+'bookresult.json', 'w')

    def close_spider(self, spider):
        print('JSON File Generated')
        self.file.close()

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line)
        return item


import logging
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor


class BooksCrawlSpider(CrawlSpider):
    name = 'books-crawlspider'
    allowed_domains = ['toscrape.com']
    start_urls = ['http://books.toscrape.com']
    custom_settings = {
      'LOG_LEVEL': logging.WARNING,
      'ITEM_PIPELINES': {'__main__.ConsoleWriterPipeline': 1}#, # Used for pipeline
      }
    rules = [
        Rule(
            LinkExtractor(allow=('/catalogue/page-\d+.html')),follow=True
        ),
        Rule(
             LinkExtractor(deny=('/category/books', '.com/index.html')),callback='p
            follow=True
        ),
    ]

    def parse_book_page(self, response):
        yield {
            'title': response.css('.product_main h1::text').get(),
            'price': response.css('.product_main p.price_color::text').re_first('£(
            'stock': int(
                ''.join(
                    response.css('.product_main .instock.availability ::text').re('
                )
            ),
        }

#  uncomment me and run
from scrapy.crawler import CrawlerProcess

hgw_crawler_process = CrawlerProcess({
    'USER_AGENT': 'Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1)'
})

hgw_crawler_process.crawl(BooksCrawlSpider)
hgw_crawler_process.start()
```

```
    {"title": "The Autobiography of Malcolm X", "price": "23.43", "stock": 3}

    {"title": "The Book Thief", "price": "53.49", "stock": 3}

    {"title": "Shopaholic & Baby (Shopaholic #5)", "price": "46.45", "stock": 3}

    {"title": "Quiet: The Power of Introverts in a World That Can't Stop Talking",

    {"title": "One for the Money (Stephanie Plum #1)", "price": "32.87", "stock":

    {"title": "Packing for Mars: The Curious Science of Life in the Void", "price"

    {"title": "Orange Is the New Black", "price": "24.61", "stock": 3}

    {"title": "Morning Star (Red Rising #3)", "price": "29.40", "stock": 3}

    {"title": "Lean In: Women, Work, and the Will to Lead", "price": "25.02", "stc

    {"title": "Life After Life", "price": "26.13", "stock": 3}

    {"title": "Gone Girl", "price": "37.60", "stock": 3}

    {"title": "Is Everyone Hanging Out Without Me? (And Other Concerns)", "price":

    {"title": "Dracula", "price": "52.62", "stock": 3}

    {"title": "Dark Places", "price": "23.90", "stock": 3}
```

```
{"title": "David and Goliath: Underdogs, Misfits, and the Art of Battling Giar

{"title": "Dead Wake: The Last Crossing of the Lusitania", "price": "39.24", "

{"title": "Eclipse (Twilight #3)", "price": "18.74", "stock": 3}

{"title": "Fellside", "price": "38.62", "stock": 3}

{"title": "Breaking Dawn (Twilight #4)", "price": "35.28", "stock": 3}

{"title": "Beautiful Creatures (Caster Chronicles #1)", "price": "21.55", "stc

{"title": "A Visit from the Goon Squad", "price": "14.08", "stock": 3}

{"title": "The Zombie Room", "price": "19.69", "stock": 1}

{"title": "The Name of the Wind (The Kingkiller Chronicle #1)", "price": "50.5

{"title": "Taking Shots (Assassins #1)", "price": "18.88", "stock": 1}

{"title": "Paradise Lost (Paradise #1)", "price": "24.96", "stock": 1}

{"title": "Shatter Me (Shatter Me #1)", "price": "42.40", "stock": 1}

{"title": "Jane Eyre", "price": "38.43", "stock": 1}

{"title": "On the Road (Duluoz Legend)", "price": "32.36", "stock": 1}

{"title": "Frankenstein", "price": "38.00", "stock": 1}
```

RestFul API

Let's use API from data.gov.sg to check the PSI readings.

https://api.data.gov.sg/v1/environment/psi

When you click on the above link, you see that the data return is in json format.

With reference to lecture slide 34 and 35, let's extract the PSI 24-hourly reading.

```
# importing the requests library
import requests
# api-endpoint
URL = "https://api.data.gov.sg/v1/environment/psi"

# sending get request and saving the response as response object
r = requests.get(url = URL)


# extracting data in json format
data = r.json()
print(data)
```

```
{'region_metadata': [{'name': 'west', 'label_location': {'latitude': 1.35735,
```

It is quite difficult to view the json data from the notebook. We can make use of online JSON Viewer such as http://jsonviewer.stack.hu/ to help us.

From the json viewer, look for psi_twenty_four_hourly which is the data that we want to display.

Note that the sample in lecture is reading pm25 one hourly reading, but now we want PSI 24-hourly reading.

```
# extracting PSI24 readings
readings = data['items'][0]['readings']['psi_twenty_four_hourly']
print("The PSI 24hourly readings are")
for r in readings:
  print("%s : %s"%(r, readings[r]))
```

```
The PSI 24hourly readings are
west : 36
national : 54
east : 52
central : 54
south : 33
north : 51
```