A Classy Spider

WEB SCRAPING IN PYTHON

Thomas Laetsch
Data Scientist, NYU

Your Spider

```
import scrapy
from scrapy.crawler import CrawlerProcess

class SpiderClassName(scrapy.Spider):
    name = "spider_name"
    # the code for your spider
    ...

process = CrawlerProcess()

process.crawl(SpiderClassName)

process.start()
```

Your Spider

- Required imports

```
import scrapy
from scrapy.crawler import CrawlerProcess
```

- The part we will focus on: the actual spider

```
class SpiderClassName(scrapy.Spider):
    name = "spider_name"
    # the code for your spider
    ...
```

- Running the spider

```
# initiate a CrawlerProcess
process = CrawlerProcess()

# tell the process which spider to use
process.crawl(YourSpider)

# start the crawling process
process.start()
```

Weaving the Web

```
class DCspider( scrapy.Spider ):

    name = 'dc_spider'

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )

    def parse( self, response ):
        # simple example: write out the html
        html_file = 'DC_courses.html'
        with open( html_file, 'wb' ) as fout:
            fout.write( response.body )
```

- Need to have a function called `start_requests`
- Need to have at least one parser function to handle the HTML code

We'll Weave the Web Together

WEB SCRAPING IN PYTHON

A Request for Service

WEB SCRAPING IN PYTHON

Thomas Laetsch
Data Scientist, NYU

Spider Recall

```
import scrapy
from scrapy.crawler import CrawlerProcess

class SpiderClassName(scrapy.Spider):
    name = "spider_name"
    # the code for your spider
    ...

process = CrawlerProcess()

process.crawl(SpiderClassName)

process.start()
```

Spider Recall

```
class DCspider( scrapy.Spider ):
    name = "dc_spider"

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )

    def parse( self, response ):
        # simple example: write out the html
        html_file = 'DC_courses.html'
        with open( html_file, 'wb' ) as fout:
            fout.write( response.body )
```

The Skinny on start_requests

```
def start_requests( self ):
    urls = ['https://www.datacamp.com/courses/all']
    for url in urls:
        yield scrapy.Request( url = url, callback = self.parse )
```

```
def start_requests( self ):
    url = 'https://www.datacamp.com/courses/all'
    yield scrapy.Request( url = url, callback = self.parse )
```

- `scrapy.Request` here will fill in a response variable for us
- The `url` argument tells us which site to scrape
- The `callback` argument tells us where to send the response variable for processing

Zoom Out

```
class DCspider( scrapy.Spider ):
    name = "dc_spider"

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )

    def parse( self, response ):
        # simple example: write out the html
        html_file = 'DC_courses.html'
        with open( html_file, 'wb' ) as fout:
            fout.write( response.body )
```

End Request

WEB SCRAPING IN PYTHON

Move Your Bloomin' Parse

WEB SCRAPING IN PYTHON

Thomas Laetsch
Data Scientist, NYU

datacamp

Once Again

```
class DCspider( scrapy.Spider ):
    name = "dcspider"

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )

    def parse( self, response ):
        # simple example: write out the html
        html_file = 'DC_courses.html'
        with open( html_file, 'wb' ) as fout:
            fout.write( response.body )
```

You Already Know!

```
def parse( self, response ):
    # input parsing code with response that you already know!
    # output to a file, or...
    # crawl the web!
```

DataCamp Course Links: Save to File

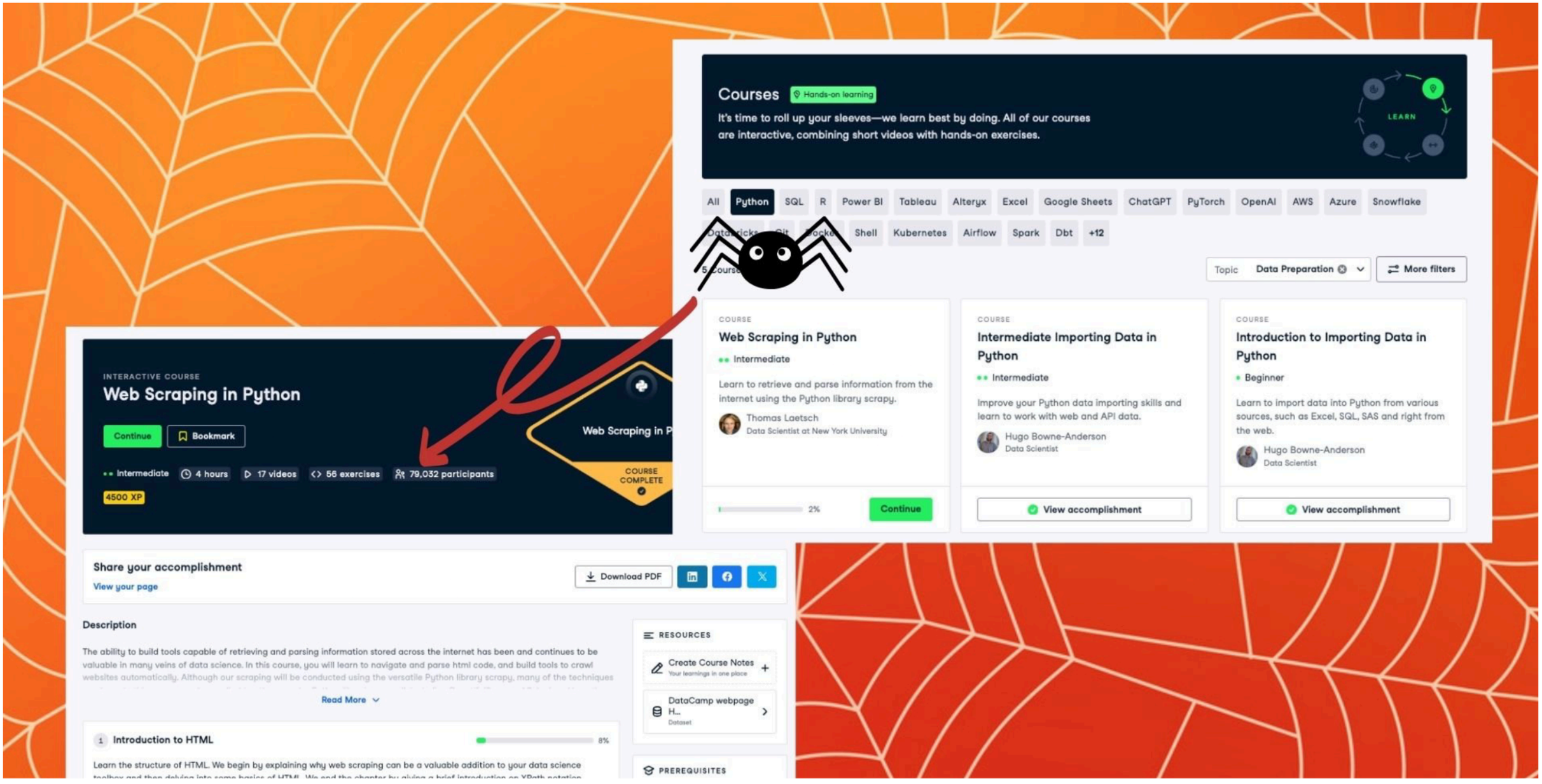
```
class DCspider( scrapy.Spider ):
    name = "dcspider"

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )
    def parse( self, response ):
        links = response.css('div.course-block > a::attr(href)').extract()
        filepath = 'DC_links.csv'
        with open( filepath, 'w' ) as f:
            f.writelines( [link + '/n' for link in links] )
```

DataCamp Course Links: Parse Again

```
class DCspider( scrapy.Spider ):
    name = "dcspider"

    def start_requests( self ):
        urls = [ 'https://www.datacamp.com/courses/all' ]
        for url in urls:
            yield scrapy.Request( url = url, callback = self.parse )
    def parse( self, response ):
        links = response.css('div.course-block > a::attr(href)').extract()
        for link in links:
            yield response.follow( url = link, callback = self.parse2 )
    def parse2( self, response ):
        # parse the course sites here!
```


Courses
Hands-on learning
It's time to roll up your sleeves—we learn best by doing. All of our courses are interactive, combining short videos with hands-on exercises.
LEARN
All
Python
SQL
R
Power BI
Tableau
Alteryx
Excel
Google Sheets
ChatGPT
PyTorch
OpenAI
AWS
Azure
Snowflake
Shell
Kubernetes
Airflow
Spark
Dbt
+12
Topic
Data Preparation
More filters
COURSE
Web Scraping in Python
Intermediate
Learn to retrieve and parse information from the internet using the Python library scrapy.
Thomas Laetsch
Data Scientist at New York University
2%
Continue
COURSE
Intermediate Importing Data in Python
Intermediate
Improve your Python data importing skills and learn to work with web and API data.
Hugo Bowne-Anderson
Data Scientist
View accomplishment
COURSE
Introduction to Importing Data in Python
Beginner
Learn to import data into Python from various sources, such as Excel, SQL, SAS and right from the web.
Hugo Bowne-Anderson
Data Scientist
View accomplishment
INTERACTIVE COURSE
Web Scraping in Python
Continue
Bookmark
Intermediate
4 hours
17 videos
56 exercises
79,032 participants
4500 XP
COURSE COMPLETE
Share your accomplishment
View your page
Download PDF
Description
The ability to build tools capable of retrieving and parsing information stored across the internet has been and continues to be valuable in many veins of data science. In this course, you will learn to navigate and parse html code, and build tools to crawl websites automatically. Although our scraping will be conducted using the versatile Python library scrapy, many of the techniques
Read More
RESOURCES
Create Course Notes
Your learnings in one place
DataCamp webpage H...
Dataset
PREREQUISITES
1 Introduction to HTML
8%
Learn the structure of HTML. We begin by explaining why web scraping can be a valuable addition to your data science

Johnny Parsin'

WEB SCRAPING IN PYTHON

Capstone

WEB SCRAPING IN PYTHON

Thomas Laetsch
Data Scientist, NYU

Inspecting Elements

```
import scrapy
from scrapy.crawler import CrawlerProcess

class DC_Chapter_Spider(scrapy.Spider):

    name = "dc_chapter_spider"

    def start_requests( self ):
        url = 'https://www.datacamp.com/courses/all'
        yield scrapy.Request( url = url,
                              callback = self.parse_front )

    def parse_front( self, response ):
        ## Code to parse the front courses page

    def parse_pages( self, response ):
        ## Code to parse course pages
        ## Fill in dc_dict here

dc_dict = dict()

process = CrawlerProcess()
process.crawl(DC_Chapter_Spider)
process.start()
```

Parsing the Front Page

```
    def parse_front( self, response ):
        # Narrow in on the course blocks
        course_blocks = response.css( 'div.course-block' )
        # Direct to the course links
        course_links = course_blocks.xpath( './a/@href' )
        # Extract the links (as a list of strings)
        links_to_follow = course_links.extract()
        # Follow the links to the next parser
        for url in links_to_follow:
            yield response.follow( url = url,
                                   callback = self.parse_pages )
```

Parsing the Course Pages

```
def parse_pages( self, response ):
    # Direct to the course title text
    crs_title = response.xpath('//h1[contains(@class,"title")]/text()')
    # Extract and clean the course title text
    crs_title_ext = crs_title.extract_first().strip()
    # Direct to the chapter titles text
    ch_titles = response.css( 'h4.chapter__title::text' )
    # Extract and clean the chapter titles text
    ch_titles_ext = [t.strip() for t in ch_titles.extract()]
    # Store this in our dictionary
    dc_dict[ crs_title_ext ] = ch_titles_ext
```

It's time to Weave

WEB SCRAPING IN PYTHON

Stop Scratching and Start Scraping!

WEB SCRAPING IN PYTHON

Thomas Laetsch
Data Scientist, NYU

Feeding the Machine

Scraping Skills

- **Objective:** Scrape a website computationally
- **How**? We decide to use `scrapy`
- **How**? We need to work with:
  - `Selector` and `Response` objects
  - Maybe even create a Spider
- **How**? We need to learn XPath or CSS Locator notation
- **How**? Understand the structure of HTML

What'd'ya Know?

- Structure of HTML
- XPath and CSS Locator notation
- How to use `Selector` and `Response` objects in `scrapy`
- How to set up a spider
- How to scrape the web

EOT

WEB SCRAPING IN PYTHON