Web Scraping

* First you need to activate your environment.
  + .\crawlenv\Scripts\activate
* For the first time use
  + pip install scrapy
  + scrapy startproject <name of the project>
  + inside the spiders folder create a new python file
  + import scrapy
  + create spider class
    - class PostsSpider(scrapy.Spider):
      * name = “posts”
      * start\_urls = [
        + ‘https://blog.scrapinghub.com/page/1’,
        + ‘https://blog.scrapinghub.com/page/2’
      * ]
      * Def parse(self, response):
        + page = response.url.split(‘/’)[-1]
        + filename = ‘posts-%s.html’ % page
        + with open(filename, ‘wb’) as f:

f.write(response.body)

* + YOU NEED TO CD INTO THE <name of the project folder> - cd bot
  + scrapy crawl posts [name of the spider class > posts]
  + –
  + –
  + –
  + Now work starts from shell in VS code
  + scrapy shell <https://blog.scrapinghub.com/> [domain name/URL of the page you want to crawl]
  + when the shell appears
    - response.css(‘title’)
    - response.css(‘title’).get()
    - response.css(‘title::text’).get()/.getall()
    - response.css(‘h3::text’).get()
    - response.css(‘h3::text’)[1].get() #gives you the second one
    - response.css(‘h2::text’)getall() #gives you all the h2 tags
    - - ------ - - - See developer tool to understand the html that are being called to extract the informations----- -\* -- -
    - response.css(‘.post-header’).getall() #all post headers
    - response.css(‘.post-header a’).get() #first link in the post header
    - response.css(‘p::text’).re(r‘scraping’) #format r string [all instances of scraping will come]
    - response.css(‘p::text’).re(r‘(\w+) you (\w+)’) #will give every word before and after the word ‘you’
    - response.css(‘p::text’).re(r‘s\w+’) #will give every word that starts with ‘s’
    - af
  + still working in the shell (part 2)
    - looping through css div to fetch title, date and author
      * for post in response.css(‘div.post-item’)
        + title = post.css(‘.post-header h2 a::text’)[0].get()
        + date = post.css(‘.post-header a::text’)[1].get()
        + title = post.css(‘.post-header h2 a::text’)[2].get()
        + print(dict(title=title, date=date, author=author))
    - **Now back to python file**
    - start\_urls will be only root url
    - changes made to **bot\_spider.py**
      * import scrapy
      * class BotSpider(scrapy.Spider):
        + name = “posts”
        + .
        + start\_urls = [

‘https://blog.scrapinghub.com/’

* + - * + ]
        + def parse(self, response):

for post in response.css(‘div.post-item’):

yield{

‘title’ : post.css(‘.post-header h2 a::text’)[0].get(),

‘date’ : post.css(‘.post-header a::text’)[1].get(),

‘author’ : post.css(‘.post-header a::text’)[2].get()

}

.

* + - * + .
    - Back to VS shell and cd into bot
    - Output in a json file
      * scrapy crawl posts –o posts.json [-o is for output file]
      * .
    - Going to a link embedded inside a page
      * def parse(self, response):
        + for post in response.css(‘div.post-item’):

yield{

‘title’ : post.css(‘.post-header h2 a::text’)[0].get(),

‘date’ : post.css(‘.post-header a::text’)[1].get(),

‘author’ : post.css(‘.post-header a::text’)[2].get()

}

.

* + - * + next\_page = response.css(‘a.next-posts-link:: attr(href)’).get()
        + if next\_page is not None:

next\_page = response.urljoin(next\_page)

yield scrapy.Request(next\_page, callback-self.parse)

* + - * + .
    - Back to VS shell cd into bot
    - Delete posts.json and run the following
      * scrapy crawl posts –o posts.json
      * .
      * .
  + .
  + .
  + .
  + .
  + .
* Asdf
* Asdf
* Asf
* asfd