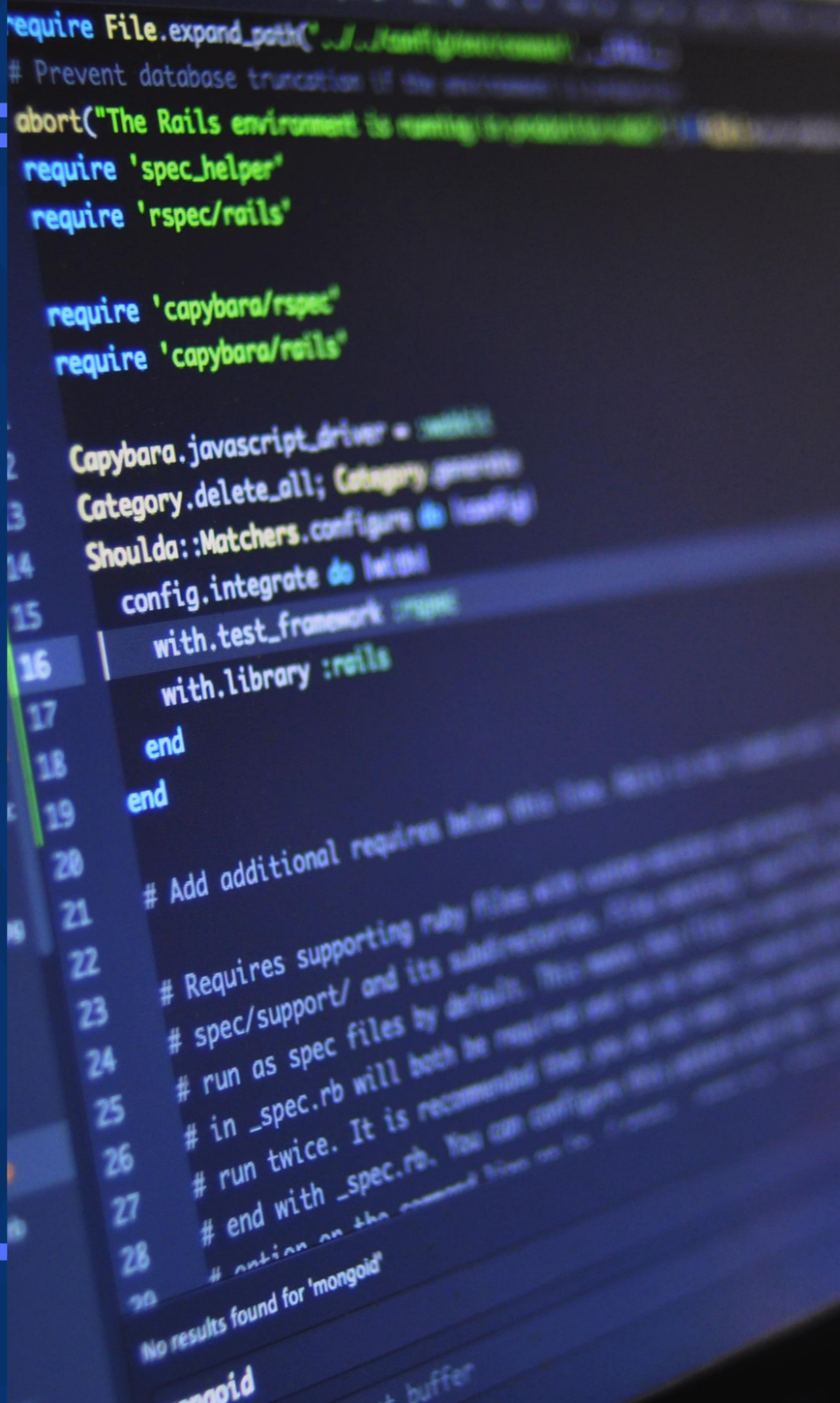
Mid-Term Assignment

Scrapy

Mrityunjay Misra
MSBA'24



Introduction - Scrapy

Scrapy, the ***web-crawling*** framework in Python, was created by a developer named ***Pablo Hoffman***. He developed Scrapy while working at Scrapinghub, which focuses on web scraping solutions and services. Scrapy is a ***free tool*** in Python for getting information from the internet. The framework has gained popularity for its efficiency in web scraping, handling large-scale tasks, and providing a versatile platform for data extraction from the web.

Steps To Setup Scrapy and Spider

1. Finding out which website to scrape and relevant information you want from the website
2. Installing the necessary Library in Pycharm/VS code or any tool
3. Setting up the Project, file architecture, and virtual environment in Pycharm
4. Setting up your spider
5. Following Coding Standards and guidelines
6. Once the data is scrapped push that data into JSON/XML/CSV
7. Push the Code to the Githb

What Needs To Be scraped ?

Box Office Mojo
by IMDbPro

It’s a Movie Ranking website by IDMB Pro which gives ranking based on many parameters.

Below are the ***cases*** in which I have used ***Scrapy*** and crawled the data-

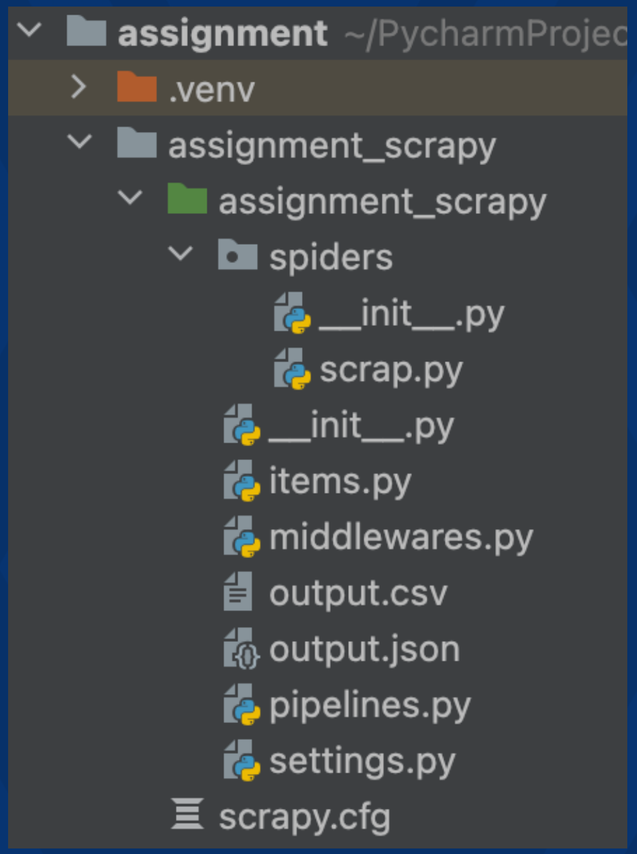
1. Title of website, All Title of Top Movie and its Ranking.
2. Box office Collection of all Movies Domestic and International
3. By Release Year and Region of Avtaar Movie
4. All Movie Released in 2009
5. Saving them to JSON and CSV Files

Webiste URL - https://www.boxofficemojo.com/chart/top_lifetime_gross/?area=XWW

Starting Project In Pycharm

Setting File Structure and Spider

1. Go to File> Create new project
2. Pycharm Automatically creates a Virtual Environment- What it Does > It’s an isolated unit if you install the scrapy package then it will only work in this environment and not outside any files
3. Go to Settings > Project interpreter r> Install Scrapy
4. Open your terminal, it will directly take you to your scrapy file (venv) > add- scrapy start project Assignment. (Assignment is the name of the spider)
5. Once the above process is done, the file structure will be created as shown in the screenshot.
6. Spider - It’s the Python program that scrapes the website
7. So Now we will be writing our Python code inside the spider file> create a folder scrapy.py to perform your code



```python
import scrapy
class BoxOfficeCollection(scrapy.Spider):
    #name of scrapy
    name = 'hollywood'
    start_urls = [
        'https://www.boxofficemojo.com/chart/top_lifetime_gross/?area=XWW',
        'https://www.boxofficemojo.com/title/tt0499549/credits/?ref_=bo_tt_tab#tabs',
        'https://boxofficemojo.com/title/tt0499549/?ref_=bo_tt_tab#tabs',
        'https://www.boxofficemojo.com/year/world/2009/?ref_=bo_cso_table_1'
    ]
    #response = source code( takene from HTML of website to get specfic locaiton to scrap
```

1. Import scrapy package
2. class BoxOfficeCollection(scrapy.Spider): > I call the class as BoxOfficeCollection and Scrapy. spider means that this class will inherit from scrapy and scrapy will inherit from a spider.
3. Create a variable, the Name of our spider ‘Hollywood’
4. scrapy needs a URL to scrap, so I have analyzed 5 cases on different links of the same website and added 4 links re-directing to different web pages.

```
def parse(self, response):
    #Below are the names of the scraped items of website
    #css selector - where if condition used with css and title text
    title = response.css('title::text').extract()
    Rank = response.css('a.a-link-normal::text').extract()
    Revenue = response.css('td.mojo-field-type-money::text').extract()
    combine = response.css('a.a-link-normal::text, td.mojo-field-type-money::text').extract()
    cast_crew =  response.css('a.a-link-normal::text').extract()
    release_region_avtaar = response.css('th::text,td::text').extract()
```

5. Now we need to create a Method called parse- It requires 2 things - self instance/self-reference and response which contains the source code of our website which we want to scrap
6. title = response.css('title::text').extract() - We are asking to scrapy to go to this source code find the title tag and extract the title tag, then yield/return it and show it to us in the dictionary and every dictionary contains keys and values.
7. Same goes for Rank, revenue, combine (revenue and rank) , cast_crew and release region of avtaar movie and yield all the result

```
    # yield result of the above css selectors statements
    yield {
        'titletext': title, #gettig Title of movies
        'ranktext': Rank, #getting top ranked movies
        'revenuetext': Revenue, #Getting top box offlice collection
        'movieyeartext': combine, #Getting movie, year and revenue
        'cast_crew': cast_crew, #Getting detail of cast and crew of Avtaar movie
        'release_region_avtaar': release_region_avtaar, #demogrpahic detail of avtaar movie
    }
```

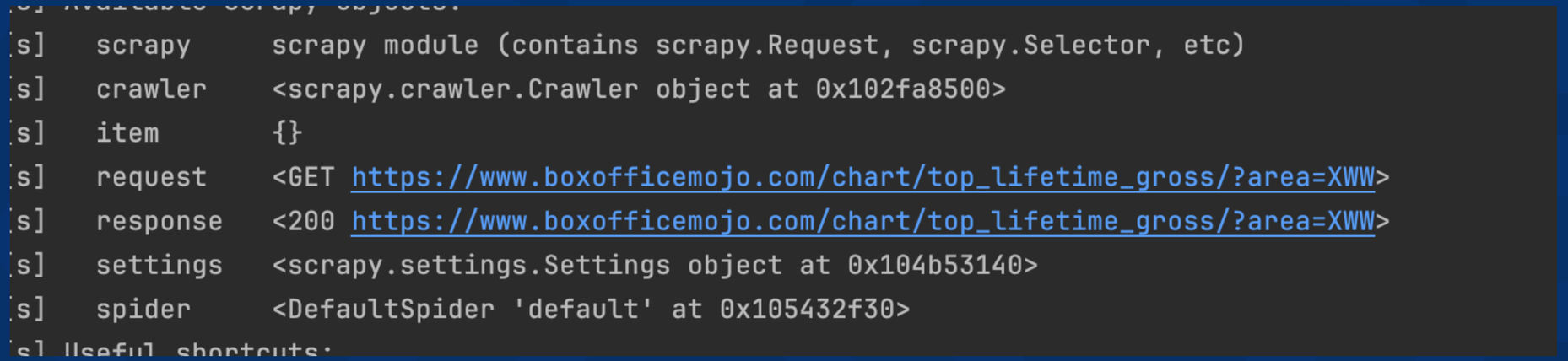
Terminal

(.venv) (base) mrityunjay@Mrityunjays-MacBook-Pro assignment % cd assignment_scrapy
(.venv) (base) mrityunjay@Mrityunjays-MacBook-Pro assignment_scrapy % scrapy crawl hollywood

##The above code in Terminal means that first we need to go to our folder by cd file name , then we need to run our crawler to scrape the complete website by running the code scrapy crawl hollwyood

(.venv) (base) mrityunjay@Mrityunjays-MacBook-Pro assignment_scrapy % scrapy shell "https://www.boxofficemojo.com/chart/top_lifetime_gross/?area=XWW"

##scrapy shell means scraping specific URL which gives response>200, means scraped successfully.



Terminal

>>> response.css("title::text").extract() ## To extract the exact title in Terminal
['Top Lifetime Grosses - Box Office Mojo'] ## This is the Exact Title

```
>>> response.css("title::text").extract()
['Top Lifetime Grosses - Box Office Mojo']
>>>
```

Now we have to inspect the source code of Avtaar movie listed on website to check the Div and class to scrape our use case , once you for the connector run the code in terminal > **response.css('a.a-link-normal::text').extract()**. ## Gives out Top movie with year

```
<a class="a-link-normal" href="/?ref_=bo_nb_cso_mojologo"></a>
</div>
▼<div class="a-section a-spacing-none mojo-nav-elements mojo-flex mojo-flex-h mojo
▶<div class="a-section a-spacing-none mojo-search-bar mojo-flex mojo-flex-h">…<
▶<div class="a-section a-spacing-none mojo-mobile-options">…</div> == $0
▼<div class="a-section a-spacing-none mojo-options mojo-flex">
▼<div class="a-popover-preload" id="a-popover-mojoRollover">
▶<a class="a-link-normal mojo-rollover-image" href="https://pro.imdb.com/logi
cso_rollover&rf=mojo_nb_cso_rollover">…</a>
</div>
```

Terminal

##Repeat the Process for all the use cases, just one shifting to a new link use scrapy shell to scrape that specific URL

>>>response.css('td.mojo-field-type-money::text').extract() - ## Scraping all the revenue of all listed movies

##Case 2
>>>response.css('a.a-link-normal::text, td.mojo-field-type-money::text').extract()
##Scrpaing movie, year and revenue combined

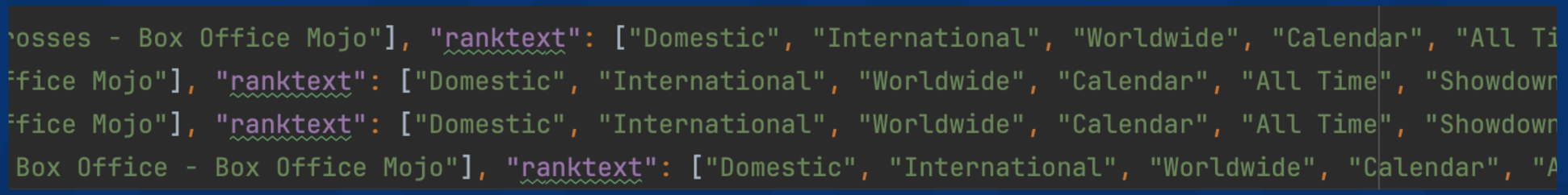
##Case 3
>>> response.css('a.a-link-normal::text').extract()
##Scraping cast and crew of avatar movie
>>>response.css('th::text,td::text').extract()
##Scraping By release, region, rank data of avatar movie

##Case 4
>>>response.css('a.a-link-normal::text').extract()
Scraping all 2009 movies list

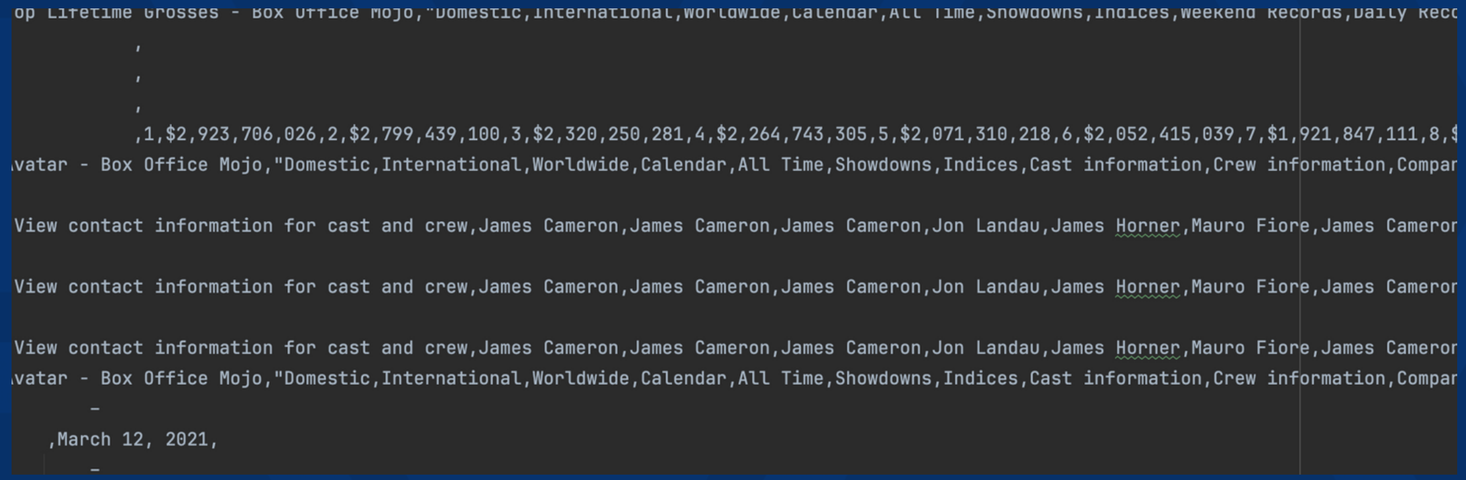
>>>scrapy crawl hollywood -o output.json/csv
##exporting all the data into json and csv (Run exit())

Once you run all the code in the terminal you will get scraped data as shown in the Terminal and CSV, JSON

Terminal



Json



CSV

Thank you!