PEGAR NO COMECO DO VIDEO O OBJETIVO DO PROJETO

First Step – Collecting Data with Scrapy

We will use scraping to collect data from the site ‘Mercado Livre’. (Scrapy will collect data in the form of html, javascript) Scrapy makes a requisition in the site and responds with HTML and will use PROCV to find each of the information we want such as banner , name and prices.

Pip install scrapy

$ scrapy startproject coleta # creates a folder with a framework with all the divisions of the web scrap project

$ cd coleta

$ scrapy genspider mercadolivre <https://lista.mercadolivre.com.br/tenis-corrida-masculino>

# scrapy genspider > and give the name of the site you want to scrap

# this spider command have 3 main goals: request (access site); parser (collect certain info like price, category) and send in it to database; and next\_page (takes to the following page)

## ONE OF THE MOST CHALLENGING THINGS TO DO WHEN USING SCRAPING IS THE PARSER WHICH IS TO LOOK FOR THE HTML INFO YOU WANT SUCH AS PRICE ETC

SO SCRAPY HAS A prompt ‘scrapy shell’ THAT FACILITATES THE JOB OF COLLECTING (PARSE)

Then you run fetch (‘https://lista.mercadolivre.com.br/tenis-corrida-masculino’)

* This will give an error because the sites require an user to access it. So we can use USER AGENT to simulate a user
* In order to do that we will :

A screenshot of a computer

Description automatically generated

And in settings.py and user agent as shown below,

A screenshot of a computer

Description automatically generated

Exit()

Scrapy shell

Fetch(“<https://lista.mercadolivre.com.br/tenis-corrida-masculino>”)

RESULT > 2024-10-01 12:27:57 [scrapy.core.engine] DEBUG: **Crawled (200**) <GET https://lista.mercadolivre.com.br/tenis-corrida-masculino> (referer: None)

# Deleted folder Middleware as we will not be using any proxy and delete pipeline folder as we will use do the transformation on pandas.

# We have made the requisition using scrapy and now it returns a response which is the HTML from the site including the info we need to collect for our analysis

>>> response

<200 <https://lista.mercadolivre.com.br/tenis-corrida-masculino>>

# Write response.text I will have access to the site`s HTML and now we can Parse which is the ProcV of the site

# We decide what we want to Parse according to our business objectives which in our case are Brand, Price and Category

# Inspecting <https://lista.mercadolivre.com.br/tenis-corrida-masculino> to have access to the page`s HTML, we are able to copy and from the HTML the codes that we want. We are parsing CONTENT for example which is <div class=”ui-search-result\_\_content”> as seen pointed out in the image below.

# In git bash written as > response.css('div.ui-search-result\_\_content')

# To check the amount of contents > len(response.css('div.ui-search-result\_\_content'))

A screenshot of a computer

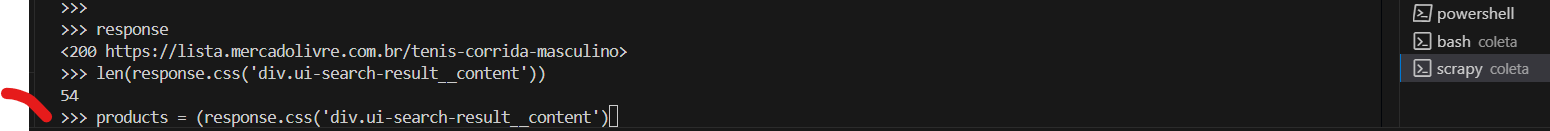
Description automatically generated

# We will create a variable “produtos” as shown below so we can iterate each one of the 54 items.

A close-up of a computer screen

Description automatically generated

# In terminal scrapy – coleta we will write > products = (response.css('div.ui-search-result\_\_content')



# Now, going for brand > <span class=’ui-search-item\_\_brand-discoverability ui-search-item\_\_group\_\_element”>

# We will write in terminal > products.css(“span.ui-search-item\_\_brand-discoverability.ui-search-item\_\_group\_\_element # if there is space we must fill in with “.”

It will return all elements 54 brands that can be found in the website. Important: We want only 1.

Using the method .get() ,

>>> products.css('span.ui-search-item\_\_brand-discoverability.ui-search-item\_\_group\_\_element').get()

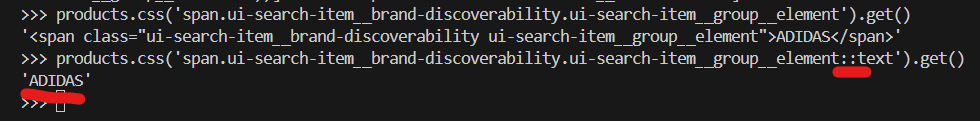
# it returns : Adidas below

'<span class="ui-search-item\_\_brand-discoverability ui-search-item\_\_group\_\_element">ADIDAS</span>'

# adding ::text in the code, it will only return us Adidas, so:

products.css('span.ui-search-item\_\_brand-discoverability.ui-search-item\_\_group\_\_element::text').get()

THE ADVANTAGE OF DOING ALL THESE STEPS IN THE TERMINAL (SCRAP SHELL) IS THAT WE CAN TEST IS BEFORE PASSING IT INTO OUR PRODUCTION PYTHON CODE



# In mercadolivre.py , we added yield as below:

A screenshot of a computer program

Description automatically generated

# We have created a folder named Data to add the created data within it. We created it by using the command:

$ scrapy crawl mercadolivre -o data.jsonl # instead of json, it could also be saved as .csv

THE BUSINESS TEAM REQUESTED THE BELOW WHICH STILL NEEDS TO BE DONE:

A white paper with black text

Description automatically generated

# For the other items we will do it directly in the project instead of testing on the terminal as we were doing on the previous one.

‘name’: product.css(‘h2.ui-search-item\_\_title::text’).get(),

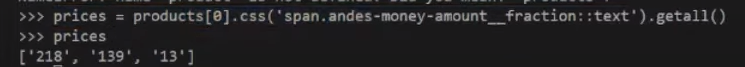
‘reviews\_rating\_number’: product.css(‘span.ui-search-reviews\_\_rating-number::text’).get(),

‘reviews\_amount’: product.css(‘span.ui-search-reviews\_\_amount::text’).get()

# For price and we will face 2 challenges that will need to be dealt with forehand. First, the price, price and cents are divided in two html code of line. (ex. 217, 39). We will run:

prices = product[0].css(‘span.andes-money-amount\_\_fraction::text’).getall()

# the result is :



# which corresponds to the amount of the 3 different prices below in red

A close up of a shoe

Description automatically generated

 for product in products:      # extracting brand from each block

           yield {

                'brand': product.css('span.ui-search-item\_\_brand-discoverability.ui-search-item\_\_group\_\_element::text').get()

                'name': product.css('h2 > a::text').get(),

                'old\_price\_reais': prices[0] if len(prices) > 0 else None, # preco riscado

                'new\_price\_reais': prices[0] if len(proces) > 0 else None,

                'reviews\_rating\_number': product.css('span.ui-search-reviews\_\_rating-number::text').get(),

                'reviews\_amount': product.css('span.ui-search-reviews\_\_amount::text').get()

           }

A screenshot of a computer

Description automatically generated

# Now run $ scrapy crawl mercadolivre -o data.csv

And you get all the desired data in a csv file like below:

A screen shot of a computer program

Description automatically generated

# saving the scraping as jsonl in a new folder called DATA to be able to better consume and manage the data retrieved. We will delete the .csv

# rename the folder ‘coleta’ to ‘source’ where we will save all the project`s code

# create a new folder called ‘ transformacao’ and one called ‘dashboard’

# We need now to finish the extraction which still needs to collect data from the next pages of the site. There are different ways that this can be done but we will go to the ‘SEGUINTE’ {next} buttom and Inspect for the html code. The code is:

Response.css(‘li.andes-pagination\_\_button.andes-pagination\_\_button—next a : : attr(href)’).get()

A screen shot of a computer program

Description automatically generated

# add IF so the it keeps extracting info from the next page until a certain point so it does not keep running indefinitely

SCRAP TOOL FOLLOWS ROBERTS.TXT WHICH IS A TEXT FILE FOUND IN THE SERVER AND ALERTS ROBOTS THAT ACCESS YOUR WEBSITE WHAT CAN BE DONE OR NOT. We will deactivate Roberts.txt

A screenshot of a computer

Description automatically generated