Alain Kiemde

March, 2023

## Contents

Figure 1: scrapy logo

## Scrapy

Scrapy is an open source Python framework used for extracting data from websites. It is very useful for collecting data at a large scale, automating scraping tasks, and performing data analysis.

## Why use scrapy?

Scrapy is a fast high-level web crawling and Web scraping framework, used to crawl websites and extract structured data from their pages.

## Installing Scrapy

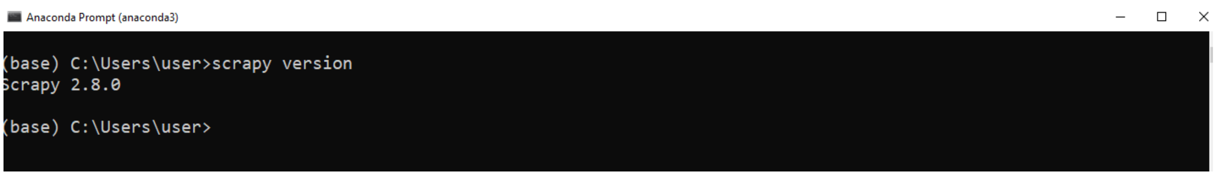
There are different ways to install scrapy. The easiest option is to install the Anaconda distribution. You can also install Scrapy using pip or conda.

## Checking the installation

To check if Scrapy is properly installed on your system, you can execute the following command in your terminal:

```
scrapy version
```

This should display the version of Scrapy installed on your system. If Scrapy is installed correctly, you should see an output similar to this:



```
(base) C:\Users\user>scrapy version
Scrapy 2.8.0

(base) C:\Users\user>
```

Figure 2: Illustration on Anaconda

## 1. Using Scrapy

### Command-line interface overview

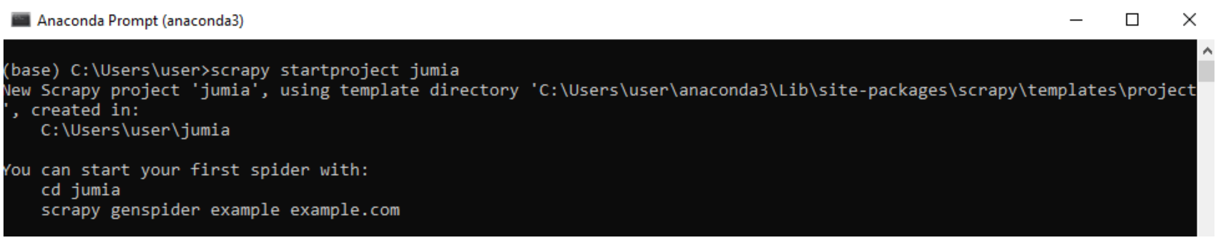
Scrapy's command-line interface is used to execute Scrapy commands to create new projects, run spiders to extract data, and more.

### Creating a new project

To create a new Scrapy project, you can use :

```
scrapy startproject project_name
```

command in Scrapy's command-line interface .For exemple :



```
(base) C:\Users\user>scrapy startproject jumia
New Scrapy project 'jumia', using template directory 'C:\Users\user\anaconda3\Lib\site-packages\scrapy\templates\project
', created in:
    C:\Users\user\jumia

You can start your first spider with:
    cd jumia
    scrapy genspider example example.com
```

Figure 3: Illustration on Anaconda

## 2. Structure of a Scrapy project

In our case, here is the structure of our Jumia project.

### spiders

The spiders stored in a folder named "spiders" are Python classes that define how to extract data from a website.

```
from import scrapy , spider
class Myspider(scrapy):
    name="myspider"
    start_urls=["http://www.example.com"]
    # Extract data from the response
    def parse(self,reponse):
        title=reponse.css("title::text").get()
    # Yield an item containing the extracted data
        yield {"title": title}
```

In this example, MySpider is a spider that extracts the title of a webpage and stores the result in an item object.

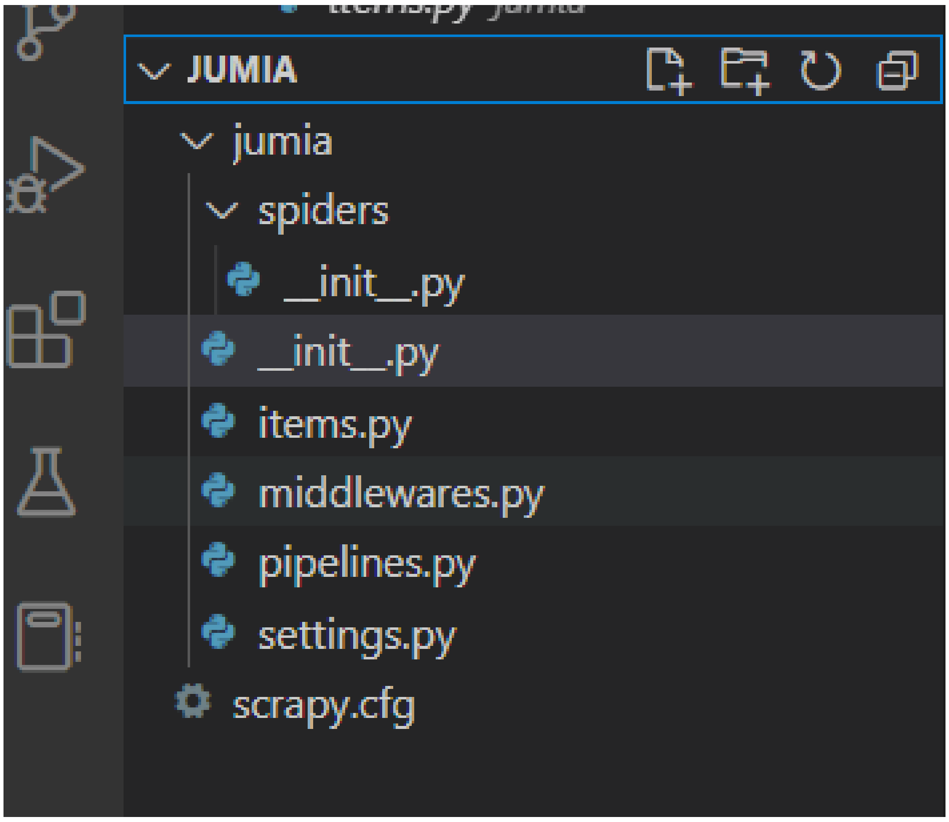
### init.py

file can be empty, or it can contain Python code that needs to be executed when the package is imported. It can contain class definitions, functions, variables, constants, module imports, and so on.

### items.py

Items are containers that store the data extracted by the spiders.

```
import scrapy
class MyItem(scrapy.Item):
     title = scrapy.Field()
     description = scrapy.Field()
```



```
JUMIA
  jumia
    spiders
      __init__.py
    __init__.py
    items.py
    middlewares.py
    pipelines.py
    settings.py
  scrapy.cfg
```

Figure 4: Opening the project on VSCode

In this example, MyItem is an item object that contains title and description fields to store data extracted by a spider.

## Middleware.py

Middleware are Python classes that provide additional functionality to Scrapy, such as filtering requests or processing responses.

```
class CustomMiddleware:
    def process_request(self, request, spider):
        # Modify the request before it is sent
        request.headers["User-Agent"]="Mozilla/5.0 (Windows NT 10.0; Win64; x64)

    def process_response(self, request, response, spider):
        # Modify the response before it is returned to the spider
        if response.status == 404:
           return scrapy.Request("http://www.example.com", callback=spider.parse_error)
        else:
            return response
```

In this example, CustomMiddleware is a class that modifies requests and responses by adding a custom User-Agent header to requests and checking the response status code. If the status code is 404, the middleware sends a new request to another page.

## Pipelines.py

Pipelines are Python classes that handle the data extracted by the spiders.

```
import json
class MyPipeline:
   def __init__(self):
       self.file = open("data.json", "w")

   def process_item(self, item, spider):
       # Write the item to a JSON file
       line = json.dumps(dict(item)) + "\n"
       self.file.write(line)
       return item

   def close_spider(self, spider):
       # Close the file when the spider is done
       self.file.close()
```

In this example, MyPipeline is a class that stores the data extracted by a spider in a JSON file. The process_item method is called for each extracted item, which is then written to the file. The close_spider method is called when the spider has finished its work, and it closes the file.

### Settings.py

Settings are configuration variables that define the behavior of Scrapy.

```
BOT_NAME = "mybot"
USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"

ROBOTSTXT_OBEY = True

ITEM_PIPELINES = {"myproject.pipelines.MyPipeline": 300}
```

In this example, Scrapy settings are defined in a settings.py file. The bot name, User-Agent header used for requests, whether to obey the robots.txt file, and the pipeline used are all specified in this file.

### Scrapy.cfg

This is the main configuration file for Scrapy. It contains information such as the project name, default settings, and spider locations.

## 3. Data extraction

In our case, we will extract data on the articles from the Jumia website (https://www.jumia.com.tn)

### Writing in Item.py

```
import scrapy
class ArticleItem(scrapy.Item):
    # Definition of fields for the ArticleItem object
    # Field for article designations
    designations=scrapy.Field()
    # Field for article images
    picture=scrapy.Field()
    # Field for article prices
    price=scrapy.Field()
```

In this code, we retrieve the designation, image, and price of an article.

## Creating a spider file in the spider folder

We will create a new file "article.py" in the spider folder. In this file, we will write our spider which is nothing but a class inheriting from the Scrapy spider class.

```
# Import necessary modules
from scrapy import Request, Spider
from ..items import ArticleItem
from jumia import items

# Create a spider class
class SpiderArticle(Spider):
    # Define spider name
    name = "article"
    # Define the URL to scrape
    url = "https://www.jumia.com.tn/"

    # Define the starting point for the spider
    def start_requests(self):
        # Send a request to the URL and specify the callback function to use
        yield Request(url=self.url, callback=self.parse_article)

    # Define the callback function to extract data from the page
    def parse_article(self, response):
        # Find all the articles on the page
        listeArticle = response.css('article.prd')
        # Loop through each article and extract relevant data
        for article in listeArticle:
            # Extract the article's designation
            designations = article.css('a.core div.name::text').extract_first()
            # Extract the article's picture
            picture = article.css('a.core img.img').attrib['data-src']
            # Extract the article's price
            price = article.css('a.core div.prc::text').extract_first()
            # Create an ArticleItem instance and fill it with data
            item = ArticleItem()
            item['designations'] = designations
            item['picture'] = picture
            item['price'] = price

            # Yield the ArticleItem instance to Scrapy for further processing
            yield item
```

## data recovery

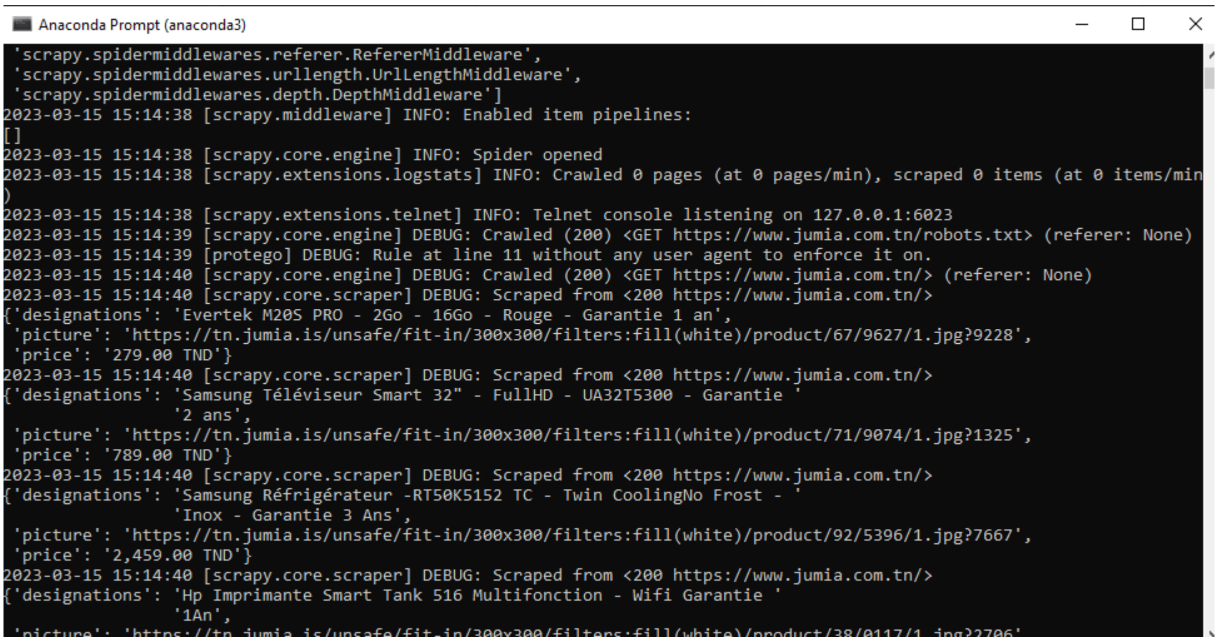
```
# This command runs the Scrapy spider called "article".
scrapy crawl article
```

the result of executing this code.

## Storing data in an Excel file named article.csv

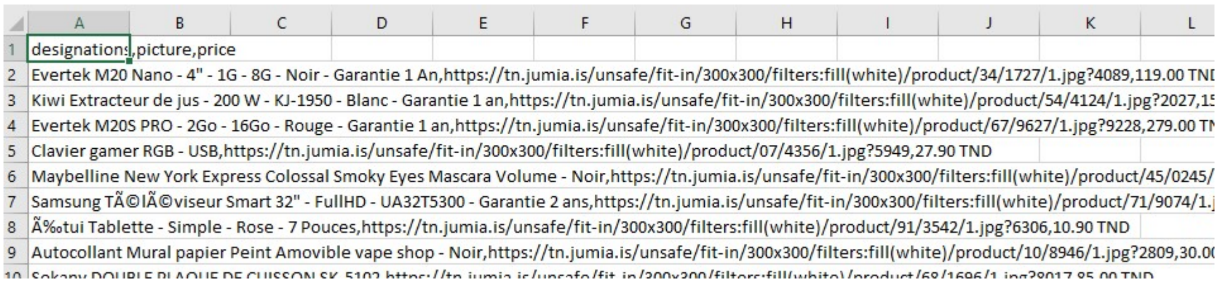
```
scrapy crawl article -o article.csv
```

the result of executing this code.



```
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2023-03-15 15:14:38 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2023-03-15 15:14:38 [scrapy.core.engine] INFO: Spider opened
2023-03-15 15:14:38 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min
)
2023-03-15 15:14:38 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2023-03-15 15:14:39 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.jumia.com.tn/robots.txt> (referer: None)
2023-03-15 15:14:39 [protego] DEBUG: Rule at line 11 without any user agent to enforce it on.
2023-03-15 15:14:40 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.jumia.com.tn/> (referer: None)
2023-03-15 15:14:40 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.jumia.com.tn/>
{'designations': 'Evertek M20S PRO - 2Go - 16Go - Rouge - Garantie 1 an',
 'picture': 'https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/67/9627/1.jpg?9228',
 'price': '279.00 TND'}
2023-03-15 15:14:40 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.jumia.com.tn/>
{'designations': 'Samsung Téléviseur Smart 32" - FullHD - UA32T5300 - Garantie '
                 '2 ans',
 'picture': 'https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/71/9074/1.jpg?1325',
 'price': '789.00 TND'}
2023-03-15 15:14:40 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.jumia.com.tn/>
{'designations': 'Samsung Réfrigérateur -RT50K5152 TC - Twin CoolingNo Frost - '
                 'Inox - Garantie 3 Ans',
 'picture': 'https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/92/5396/1.jpg?7667',
 'price': '2,459.00 TND'}
2023-03-15 15:14:40 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.jumia.com.tn/>
{'designations': 'Hp Imprimante Smart Tank 516 Multifonction - Wifi Garantie '
                 '1An',
 'picture': 'https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/38/0117/1.jpg?2706',
```

Figure 5: Confirmation of successful data extraction via the terminal



```
1  designations,picture,price
2  Evertek M20 Nano - 4" - 1G - 8G - Noir - Garantie 1 An,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/34/1727/1.jpg?4089,119.00 TNI
3  Kiwi Extracteur de jus - 200 W - KJ-1950 - Blanc - Garantie 1 an,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/54/4124/1.jpg?2027,15
4  Evertek M20S PRO - 2Go - 16Go - Rouge - Garantie 1 an,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/67/9627/1.jpg?9228,279.00 TN
5  Clavier gamer RGB - USB,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/07/4356/1.jpg?5949,27.90 TND
6  Maybelline New York Express Colossal Smoky Eyes Mascara Volume - Noir,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/45/0245/
7  Samsung TÃ©lÃ©viseur Smart 32" - FullHD - UA32T5300 - Garantie 2 ans,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/71/9074/1.j
8  Ã‰tui Tablette - Simple - Rose - 7 Pouces,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/91/3542/1.jpg?6306,10.90 TND
9  Autocollant Mural papier Peint Amovible vape shop - Noir,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/10/8946/1.jpg?2809,30.0
10 Sokany DOUBLE PLAQUE DE CUISSON SK-5102,https://tn.jumia.is/unsafe/fit-in/300x300/filters:fill(white)/product/68/1696/1.jpg?8017,85.00 TND
```

Figure 6: The data extracted in CSV format

### Displaying data with pandas.

```
import pandas as pd
df=pd.read_csv("article.csv")
df.head()
```

| Designations | Picture | Price |
|---|---|---|
| Evertek M20 Nano - 4” - 1G - 8G - Noir - Garan.. | https://tn.jumia.is/unsafe/fit-in/300x300/filt.. | 119.00 TN |
| Kiwi Extracteur de jus - 200 W - KJ-1950 - Bla... | https://tn.jumia.is/unsafe/fit-in/300x300/filt. | 159.00 TND |
| Evertek M20S PRO - 2Go - 16Go - Rouge - Garant. | https://tn.jumia.is/unsafe/fit-in/300x300/filt... | 279.00 TND |
| Clavier gamer RGB - USB | https://tn.jumia.is/unsafe/fit-in/300x300/filt. | 27.90 TND |

## 4. Using pipelines

### Extracting images using pipelines.

```
from itemadapter import ItemAdapter
from scrapy.pipelines.images import ImagesPipeline
from slugify import slugify
from scrapy import Request
import os
class CustomImagesPipeline(ImagesPipeline):
    # Pipeline for downloading images from article pages

    def get_media_requests(self, item, info):

        image_url = item['picture_url'] # use 'picture_url' instead of 'picture'
        yield Request(image_url)

    def file_path(self, request, response=None, info=None, *, item=None):

        # We limit the maximum length of the file name to 200 characters
        image_name = slugify(item.get('designations'), max_length=200)
     return f'full/{image_name}.jpg'
```

This code allows us to retrieve and download images of our articles

### Configuring our settings file to be able to run our pipelines

```
BOT_NAME = "jumia"

SPIDER_MODULES = ["jumia.spiders"]
NEWSPIDER_MODULE = "jumia.spiders"
ITEM_PIPELINES = {'jumia.pipelines.CustomImagesPipeline': 1 }
IMAGES_STORE = 'images'
# Obey robots.txt rules
ROBOTSTXT_OBEY = True
```

Updating the article.py file for image extraction.

```
from scrapy import Request, Spider
from jumia.items import ArticleItem # import ArticleItem from the correct location
from scrapy import Request
```

```
class SpiderArticle(Spider):
    name = "article"
    url = "https://www.jumia.com.tn/"

    def start_requests(self):
        yield Request(url=self.url, callback=self.parse_article)

    def parse_article(self, response):
        # Get the list of articles
        listeArticle = response.css('article.prd')
        for article in listeArticle:
            # Get the article's designations, picture URL, and price
            designations = article.css('a.core div.name::text').extract_first()
            picture_url = article.css('a.core img.img').attrib['data-src']
            price = article.css('a.core div.prc::text').extract_first()

            # Create a new ArticleItem object and assign the scraped data to its fields
            item = ArticleItem()
            item['designations'] = designations
            item['picture'] = picture_url
            item['price'] = price
            item['picture_url'] = picture_url

            # Yield the item to pass it to the pipeline for further processing
            yield item
```

Updating the items.py file for image extraction.

```
import scrapy


class ArticleItem(scrapy.Item):

      # Field for article designations
      designations=scrapy.Field()
      # Field for article images
      picture=scrapy.Field()
      # Field for article prices
      price=scrapy.Field()
      # Field for the URL of the article's picture
      picture_url = scrapy.Field()
```

Run our spider again:

```
scrapy crawl article
```

Result of the execution.

## 5.Using pipelines for data transformation

```
class CustomPricePipeline:
    # Pipeline to convert prices to euros
    exchange_rate = 0.31 # TND to EUR exchange rate
    def process_item(self, item, spider):
        # We assume the price is in TND and convert it to EUR
        tnd_price = item['price']
```

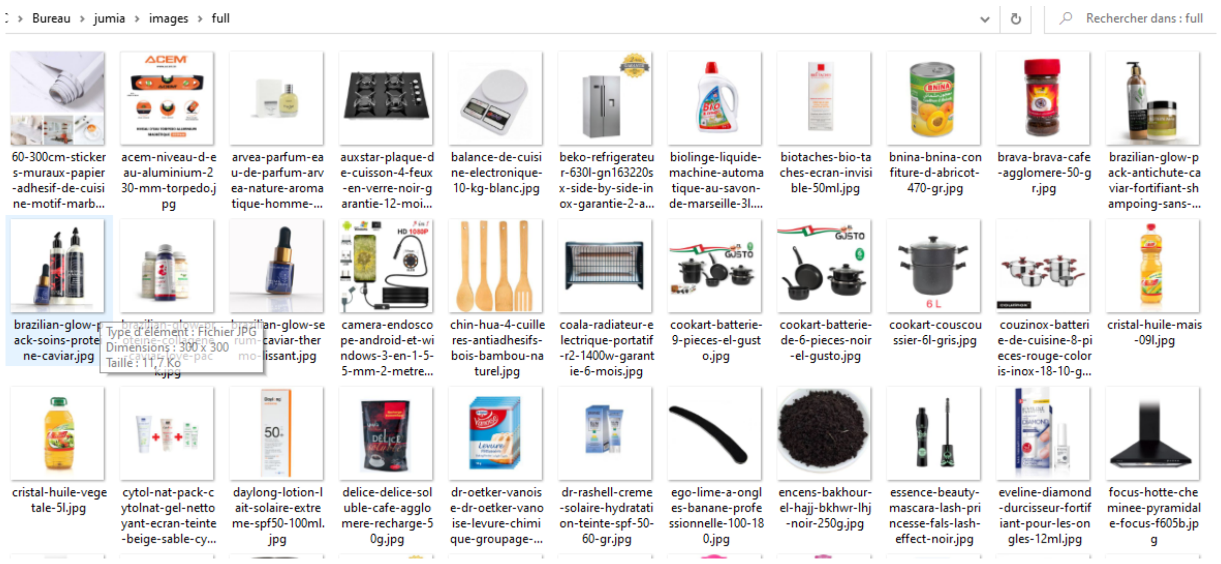

Bureau > jumia > images > full
Rechercher dans : full
60-300cm-stickers-muraux-papier-adhesif-de-cuisine-motif-marb...
acem-niveau-d-eau-aluminium-230-mm-torpedo.jpg
arvea-parfum-eau-de-parfum-arvea-nature-aromatique-homme-...
auxstar-plaque-de-cuisson-4-feux-en-verre-noir-garantie-12-moi...
balance-de-cuisine-electronique-10-kg-blanc.jpg
beko-refrigerateur-r-630l-gn163220sx-side-by-side-inox-garantie-2-a...
biolinge-liquide-machine-automatique-au-savon-de-marseille-3l....
biotaches-bio-taches-ecran-invisible-50ml.jpg
bnina-bnina-confiture-d-abricot-470-gr.jpg
brava-brava-cafe-agglomere-50-gr.jpg
brazilian-glow-pack-antichute-caviar-fortifiant-shampoing-sans-...
brazilian-glow-pack-soins-proteine-caviar.jpg
Type d'élément : Fichier JPG
Dimensions : 300 x 300
Taille : 11,7 Ko
k.jpg
brazilian-glow-se ... caviar-ther ... issant.jpg
camera-endoscope-android-et-windows-3-en-1-5-5-mm-2-metre...
chin-hua-4-cuilleres-antiadhesifs-bois-bambou-naturel.jpg
coala-radiateur-electrique-portatif-r2-1400w-garantie-6-mois.jpg
cookart-batterie-9-pieces-el-gusto.jpg
cookart-batterie-de-6-pieces-noir-el-gusto.jpg
cookart-couscoussier-6l-gris.jpg
couzinox-batterie-de-cuisine-8-pieces-rouge-coloris-inox-18-10-g...
cristal-huile-mais-09l.jpg
cristal-huile-vegetale-5l.jpg
cytol-nat-pack-cytolnat-gel-nettoyant-ecran-teinte-beige-sable-cy...
daylong-lotion-lait-solaire-extreme-spf50-100ml.jpg
delice-delice-soluble-cafe-agglomere-recharge-50g.jpg
dr-oetker-vanoise-dr-oetker-vanoise-levure-chimique-groupage-...
dr-rashell-creme-solaire-hydratation-teinte-spf-50-60-gr.jpg
ego-lime-a-ongles-banane-professionnelle-100-180.jpg
encens-bakhour-el-hajj-bkhwr-lhj-noir-250g.jpg
essence-beauty-mascara-lash-princesse-fals-lash-effect-noir.jpg
eveline-diamond-durcisseur-fortifiant-pour-les-ongles-12ml.jpg
focus-hotte-cheminee-pyramidale-focus-f605b.jpg

Figure 7: Image extraction results.

```python
        if 'TND' in tnd_price: # Check if price is in TND
            tnd_price = tnd_price.replace('TND', '').strip()
            eur_price = round(float(tnd_price) * self.exchange_rate)
            item['price'] = f'{eur_price} EUR'
        return item
```

This code converts prices to euros.

### Updating the settings.py file to run this code.

```python
BOT_NAME = "jumia"

SPIDER_MODULES = ["jumia.spiders"]
NEWSPIDER_MODULE = "jumia.spiders"
ITEM_PIPELINES = {'jumia.pipelines.CustomImagesPipeline': 1 ,
                  'jumia.pipelines.CustomPricePipeline': 2 ,

}

IMAGES_STORE = 'images'
ROBOTSTXT_OBEY = True
```

Here is the obtained result.

| Designations | Price |
|---|---|
| Evertek M20 Nano - 4” - 1G - 8G - Noir - Garan.. | 119 EUR |
| Kiwi Extracteur de jus - 200 W - KJ-1950 - Bla. | 159 EUR |
| Evertek M20S PRO - 2Go - 16Go - Rouge - Garant. | 279 EUR |

## 6. Uses of pipelines to process data.

```python
class CustomDesignationsPipeline:
    def process_item(self, item, spider):
        # Initialize an empty list to store the extracted words
        words = []

        # Loop through each designation in the 'designations' field of the item
        for designation in item['designations']:
            # Split the designation string at the '-' character and get the first element (i.e., the
            word = designation.split('-')[0].strip()
            # If the resulting word is not an empty string, append it to the 'words' list
            if word:
                words.append(word)

        # Join the words in the 'words' list into a single string, separated by spaces
        extracted_words = ' '.join(words)

        # Add the extracted words to the item dictionary
        item['extracted_words'] = extracted_words

        # Return the modified item
        return item
```

This code defines a custom pipeline class that extracts certain words from the designations field of each item processed by a spider. The extracted words are stored in a new field called extracted_words in the item dictionary.

### Updating the settings.py file to run this code.

```python
BOT_NAME = "jumia"

SPIDER_MODULES = ["jumia.spiders"]
NEWSPIDER_MODULE = "jumia.spiders"
ITEM_PIPELINES = {'jumia.pipelines.CustomImagesPipeline': 1 ,
                  'jumia.pipelines.CustomPricePipeline': 2 ,
                   'jumia.pipelines.CustomDesignationsPipeline':3
}

IMAGES_STORE = 'images'
ROBOTSTXT_OBEY = True
```

Here is the result obtained.

| Designations | Extracted_words |
|---|---|
| Samsung Téléviseur Smart 32" - FullHD - UA32T5 | S a m s u n g T é l é v i s e u r S m a r t 3 |
| Brazilian Glow Sérum Caviar Thermo-Lissant | B r a z i l i a n G l o w S é r u m C a v i a |
| Niken MINI COMPRESSEUR D AIR DE LA MARQUE NIKE... | N i k e n M I N I C O M P R E S S E U R D A I |

## conclusion

Hoping that you have learned two or three things, know that the topic is much broader than it seems. It is therefore strongly recommended that you browse the following links to perfect your learning.

## To Learn more

- Scrapy documentation

- Scrapy tutorial
- Advanced Scrapy techniques
- Scrapy vs. BeautifulSoup comparison
- Scrapy and Splash tutorial
- Scrapy and XPath tutoria
- Scrapy vs. Selenium comparison
- Scrapy architecture explained

## To go even further :

Scrapy is a powerful web scraping library, but it may not be suitable for handling extremely large datasets. When it comes to processing massive datasets, there are several alternatives to consider:

- PySpider :A powerful, open-source web crawling and web scraping framework that uses asyncio and Python's async/await syntax.
- Apache Nutch: An open-source web crawler that allows you to search and analyze web content.
- StormCrawler: An open-source collection of resources and tools for building scalable, distributed web crawlers.
- Portia: A web scraping tool that uses machine learning to help you extract data from websites without writing any code.