

Scraping Data from Amazon site and saving it in JSON format:

1. Install the scrapy library in virtual environment

```
pip install scrapy
```

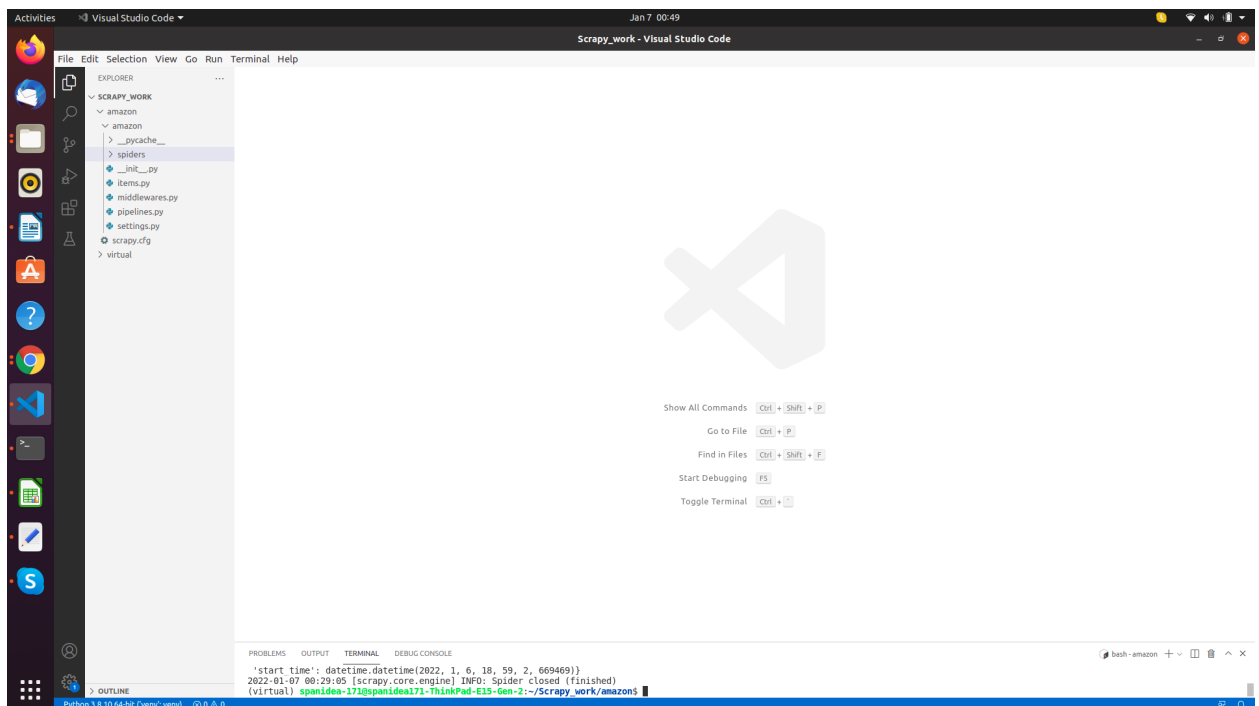
2. Create a scrapy project by using the command:

```
scrapy startproject project_name
```

In my case I have created a project with name “amazon”:

```
scrapy startproject amazon
```

This will create a project structure as shown below:



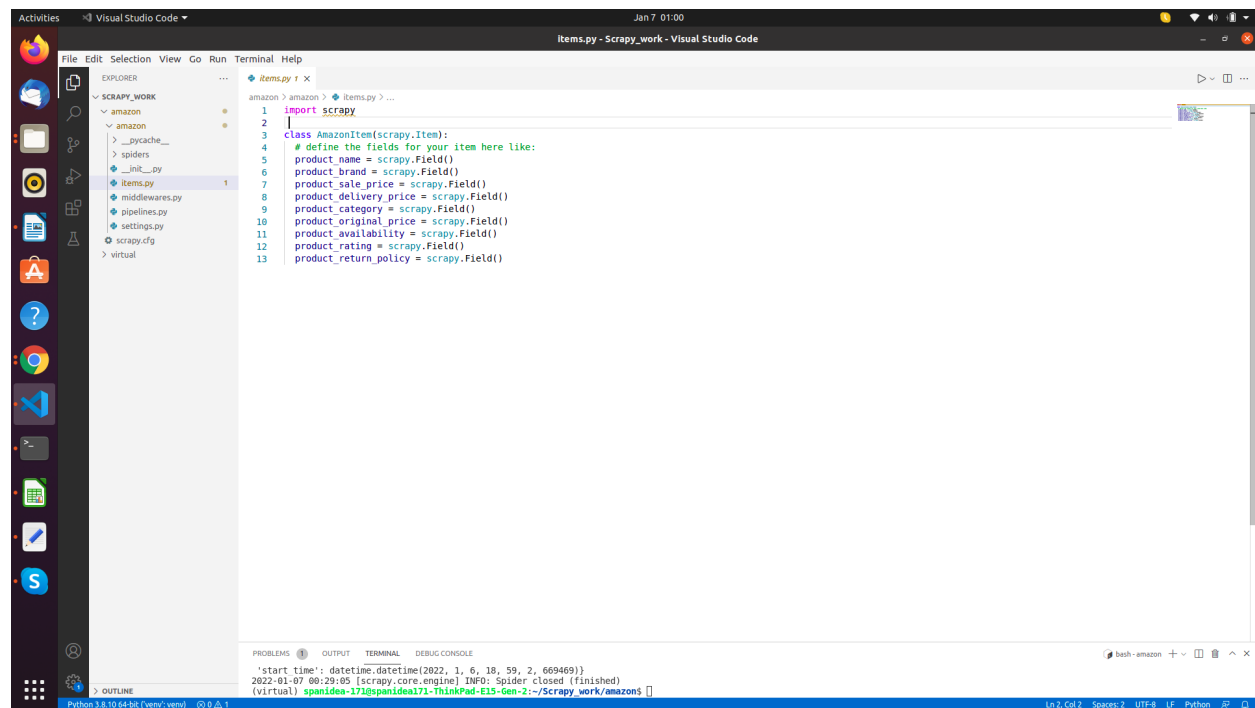
The project structure which scrapy creates for a user has,

- **scrapy.cfg:** It is a project configuration file which contains information for setting module for the project along with its deployment information.
- **amazon:** It is an application directory with many different files that are actually responsible for running and scraping data from web URLs.
- **items.py:** Items are containers that will be loaded with the scraped data. They are declared by creating a **scrapy.Item** class and defining its attributes as **scrapy.Field** objects.
- **pipelines.py:** After an item has been scraped by a spider, it is sent to the Item **Pipeline** which processes it through several components that are executed sequentially. Each item pipeline component is a Python class that has to implement a method called **process_item** to process scraped items. It receives an item and performs an action on it, also decides if the item should continue through the pipeline or should be dropped and not processed any longer.
- **settings.py:** It allows one to customize the behavior of all Scrapy **components**, including the core, extensions, pipelines, and spiders themselves.
- **spiders:** Spiders is a directory which contains all **spiders/crawlers** as Python classes. Whenever one runs/crawls any spider then scrapy looks into this directory and tries to find the spider with its name provided by the user. Spiders define how a certain site or a group of sites will be scraped, including how to perform the crawl and how to extract data from their pages.

3. The three below steps needs to be performed to scrape data:

(a) Update **items.py**: In **items.py**, declare all scrapy fields that will be used to store data from scraping. In this case, the items defined are as below:

```
product_name, product_brand, product_sale_price,  
product_delivery_price, product_category,  
product_original_price, product_availability,  
product_rating, product_return_policy.
```



(b) Create a new **Spider**: In this all required URLs needed for scraping like allowed domains, start_urls and

parse method needed to parse the scraped response object are defined. In this case, the spider is defined with name as “AmazonProductSpider”. This should be unique across all spiders.

#Name of the spider

```
name = 'AmazonProductSpider'
```

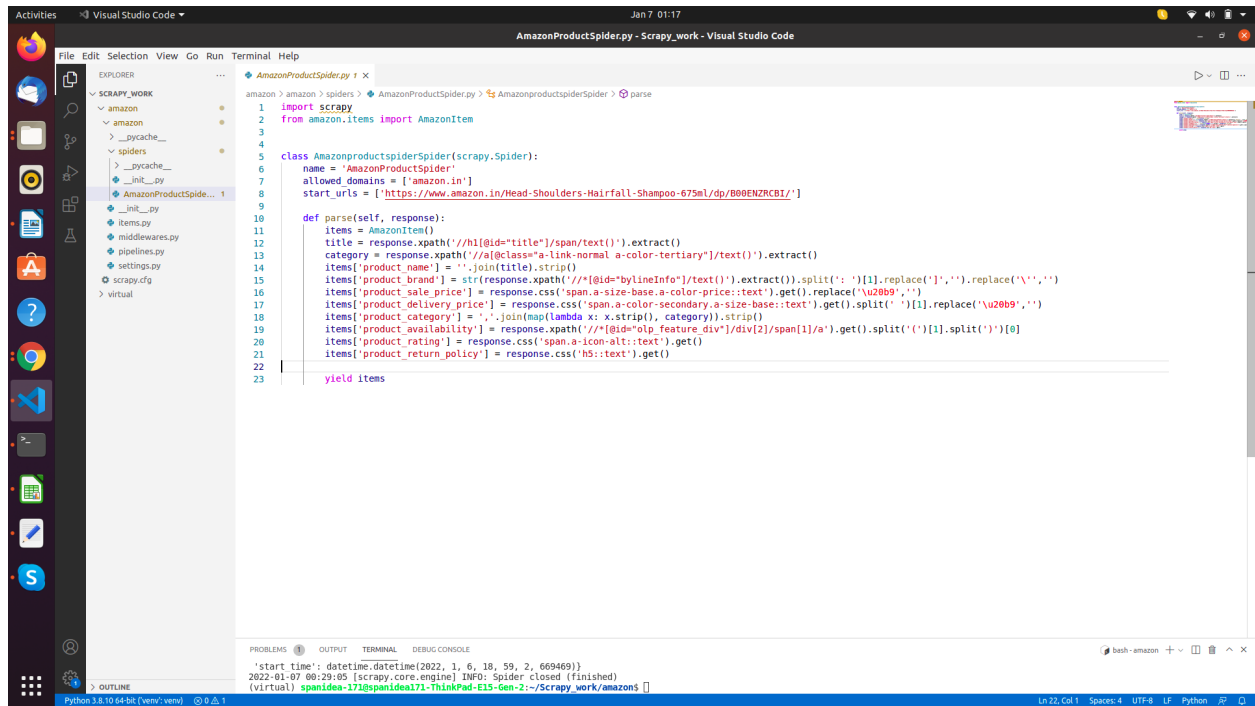
#Allowed domains

```
allowed_domains = ['amazon.in']
```

#Mention all urls which needs to be scraped

```
start_urls =  
[ 'https://www.amazon.in/Head-Shoulders-Hairfall-Shampoo-675ml/dp/  
/B00ENZRCBI/' ]
```

The code for the spider is shown below:



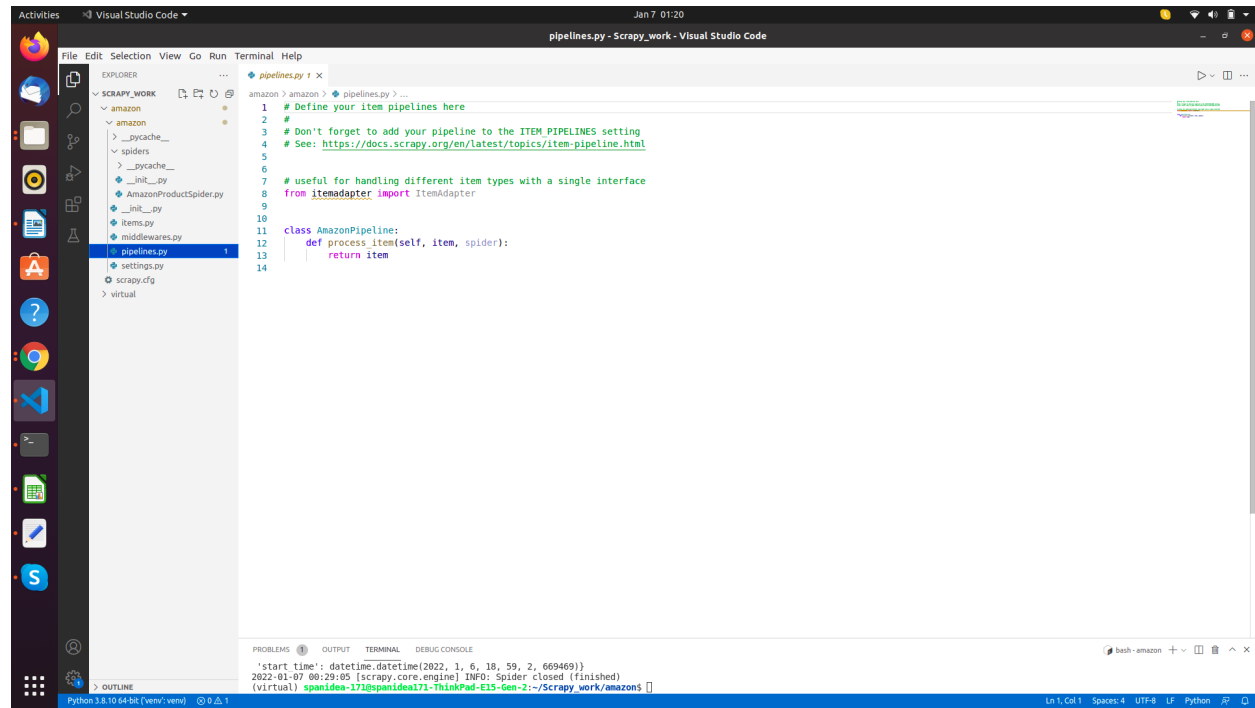
```
AmazonProductSpider.py - Scrapy_work - Visual Studio Code

1 import scrapy
2 from amazon.items import AmazonItem
3
4
5 class AmazonProductSpider(scrapy.Spider):
6     name = 'AmazonProductSpider'
7     allowed_domains = ['amazon.in']
8     start_urls = ['https://www.amazon.in/Head-Shoulders-Hairfall-Shampoo-675ml/dp/B00ENZRCBI/']
9
10    def parse(self, response):
11        items = AmazonItem()
12        title = response.xpath('//h1[@id="title"]/span/text()').extract()
13        category = response.xpath('//a[@class="a-link-normal a-color-tertiary"]/text()').extract()
14        items['product_name'] = ''.join(title).strip()
15        items['product_brand'] = str(response.xpath('//*[@id="bylineInfo"]/text()').extract()).split(':')[1].replace(' ','').replace('\n','')
16        items['product_sale_price'] = response.css('span.a-size-base.a-color-price::text').get().replace('\u20b9','')
17        items['product_delivery_price'] = response.css('span.a-color-secondary.a-size-base::text').get().split(':')[1].replace('\u20b9','')
18        items['product_category'] = ','.join(map(lambda x: x.strip(), category)).strip()
19        items['product_availability'] = response.xpath('//*[@id="olp_feature_div"]/div[2]/span[1]/a').get().split('(')[1].split(')')[0]
20        items['product_rating'] = response.css('span.a-icon-alt::text').get()
21        items['product_return_policy'] = response.css('h5::text').get()
22
23    yield items

'start time': datetime.datetime(2022, 1, 6, 18, 59, 2, 669469)}
2022-01-07 06:29:05 [scrapy.core.engine] INFO: Spider closed (finished)
(virtual) scrapy@172@spanides171-ThinkPad-E15-Gen-2-~/Scrapy_work/amazon$
```

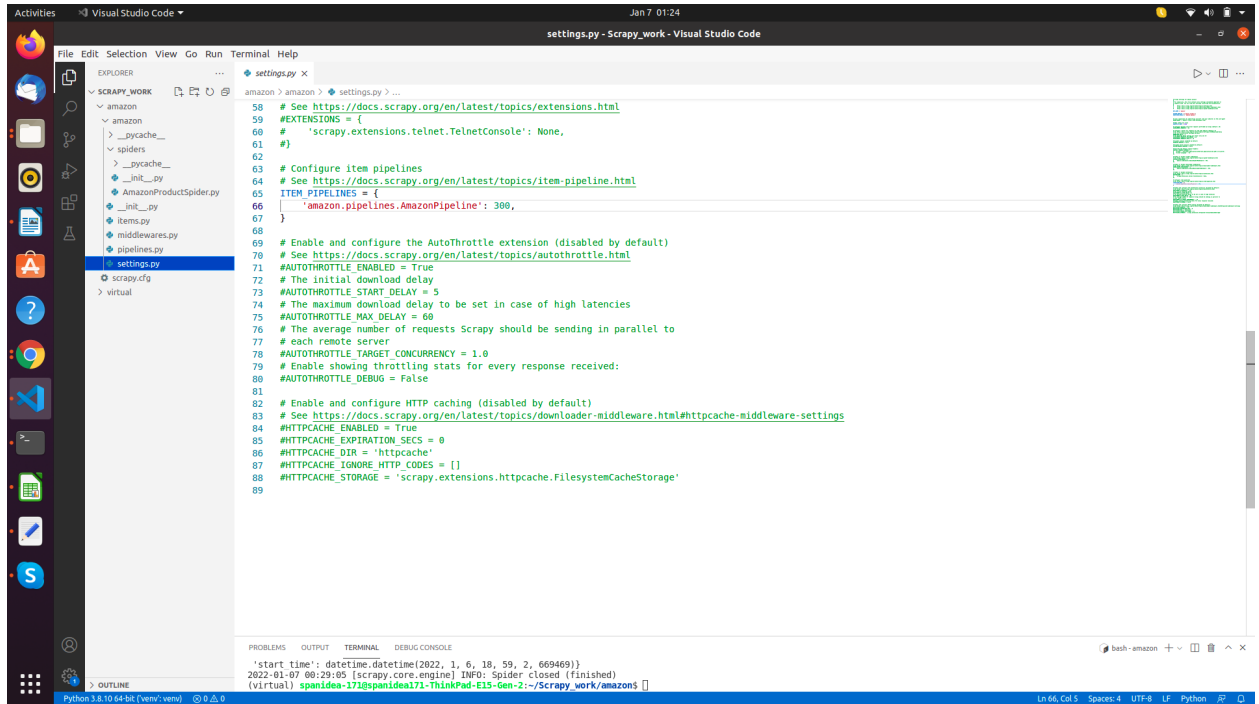
(c) Update pipeline.py if further data processing is required. In this case, the default configuration is sufficient for scraping data.

The pipeline.py is shown below:



Note: The pipeline classes used in pipeline.py needs to be enabled in settings.py.

```
ITEM_PIPELINES = {
    'amazon.pipelines.AmazonPipeline': 300,
}
```



4. For calling the spider and saving the data in item.json, the below command can be used:

```
scrapy crawl AmazonProductSpider -o items.json
```

The created items.json file created contains data in json format as shown below:

