Junior Python Scraping Challenge

The technical challenge is composed of a base exercise and a few optional improvement tasks. You can choose to do none, one, or all optional tasks as a bonus demonstration of your motivation and skills. All tasks include an estimation of how long the task should take you.

The tasks should be completed within the next 5 days after receiving it.

You can submit your solution as a link to the Git code repository to the same emails where you received this challenge.

Further questions can be addressed by email at any moment during the challenge timeframe.

The team could ask you to extend or modify the submitted solution for new requirements in a pair-programming session prior to the Technical Interview.

Implement a crawler for target.com

Estimated time: 2 - 4 hours

We want you to code a crawler capable of extracting product data from www.target.com using search keywords.

We expect a crawler capable of using the search engine of the webpage either by the HTML or the backend API. The crawler must use a set of keywords to compose the requests, navigate to the page and extract the product data of the first result page for each search keyword. We expect to receive all extracted data aggregated in the format we specify below.

Acceptance criteria:

- The solution must be implemented in Python 3.
- The crawler should be as efficient as possible (fast, low memory and CPU usage, ...).
- Documentation about how to install it & use it must be included, including how your solution expects to receive the input data and where to find the output extractions containing the results.
- Extracted data must be compliant with the data model depicted on "technical assets".
- The crawler will receive a list of keywords as input.
- The crawler will output extracted items on JSON format.
- For the purpouse of this task you only have to process first page of results.

How it will get tested/accepted

- A member of our Dev team will install & run the app from the repository on a local machine following the provided documentation.
- The team will run the solution with a prepared set of keywords, output data will be validated with the data model depicted on "technical assets" and reviewed using our own solution as reference.
- Code repository will be reviewed by the team. Any part of your submission can be a topic of discussion on the following Technical Interview.

Technical assets

Input would be a JSON as follow:

```
{
	"keywords": ["sneakers", "turkey meat", "loreal"]
}
```

More optional parameters can be added for further configuration (output file, concurrency limits, ...)

Expected **output** would be a JSON file as follow:

```
[
	{
		"url": "<https://www.target.com/p/levi-s-kids-jeffrey-501-tumbled-ul-lace-up-unisex-fashion-sneaker-shoe/-/A-83382193?preselect=83382225#lnk=sametab>",
		"title": "Levi's Kids Jeffrey 501 Tumbled UL Lace-up Unisex Fashion Sneaker Shoe",
		"image_url": "<https://target.scene7.com/is/image/Target/GUEST_799fdf82-c5e6-4755-bb97-d2566d5cdba0>",
		"price": {
			"amount": 39.99,
			"currency": "USD",
		}
	}
]
```

Attention to the data type validation:

- All URLs must be absolute URLs
- Price amount must be a Decimal
- Price currency must follow ISO 4217

For this task we suggest:

- Use pre-built libraries or frameworks for scraping, such as psf/requests, scrapinghub/web-poet or scrapy/scrapy.
- Implement data validation with models such as the ones that can be found on samuelcolvin/pydantic or schematics/schematics.

But, you are free to use / implement whatever tools of your preference.

Scrape with proxy (optional task)

Estimated time: 1 hour

We want you to extend the previously developed crawler to be able to perform requests using proxies from a given list.

With this solution we want to add an HTTP proxy direction as input, so that, each request must be routed through the proxy. Example proxies can be found on https://free-proxy-list.net.

Acceptance criteria

- Usage of proxy is optional, based on its presence on the input as parameter.
- When a proxy is provided all requests must go routed through the proxy.
- Proxy usage can be tested independently.

How it will get tested/accepted

- A member of the team will review the code.
- The implemented method will be tested independently.

Technical assets

Example **input**:

```
{
        "keywords": ["sneakers", "turkey meat", "loreal"]
        "proxy": "<http://79.143.87.138:9090>"
}
```

To ensure that you are using the proxy you can try your solution against any IP query page, such as https://www.myip.com/api-docs/.

Export extracted data to a DB (optional task)

Estimated time: 2 hours

We want you to extend the previously developed crawler to be able to export extracted data to a persistent DB for further usage.

We expect you to set up a persistent DB that fits data model needs, establish a connection and insert the output of the crawler into it.

Acceptance criteria

- Database schema should fit the data model we have already defined in the Technical Assets session of the main task.
- The crawler will connect directly to de database and output data to it instead of a file.
- Database data will be persisted between crawler executions, new data means new insertions, no cleanup required.
- Commands and instructions on how to setup the database and how to use it must be included.

How it will get tested/accepted

- A member of the team will review the code.
- The team will follow the instructions to set up the database and test the execution with the same type of data as before.

Technical assets

You are free to use any open-source DB engine of your preference. We suggest MySQL, MongoDB, or PostgreSQL.