Python фреймворк для парсинга web сайтов

Ivan Perelivskiy
ivan@aviata.me
linkedin.com/in/ivan-perelivskiy-664a2299
stackoverflow.com/users/699931/ivanperelivskiy
github.com/ivanperelivskiy/scrapy-demo-boilerplate

ФИЧИ

- Очень быстрый парсинг – запросы шедулятся и обрабатываются асинхронно.
- Встроенная поддержка для нахождения и выборки данных из HTML/XML с использованием CSS и XPath селекторов. Поддержка регулярных выражений для выборки данных.
- Широкий спектр встроенных расширений и middlewares для управления куками и сессиями, HTTP фичами, такими как компрессия, аутентификация, кэширование и т/д.

ФИЧИ

- Интерактивная консоль (с поддержкой IPython) для тестирования CSS и XPath селекторов.
- Контроль "вежливости" – download delay между запросами, ограничение количества одновременно выполняющихся запросов, auto-throttling расширение.
- Встроенная поддержка для генерации данных в форматах JSON, CSV, XML и хранения с помощью различных бэкендов FTP, S3, local filesystem.
- И много еще...

XPATH

Помимо CSS, Scrapy селекторы также поддерживают XPath выражения:

```
>>> response.xpath('//title')
[<Selector xpath='//title' data='<title>Quotes to Scrape</title>'>]
>>> response.xpath('//title/text()').extract_first()
'Quotes to Scrape'
```

- Using XPath with Scrapy Selectors
- Tutorial to learn XPath through examples
- Tutorial to learn "how to think in XPath"

PYTHON SHELL

Знакомство

this, os.path, sys, string, datetime

EXAMPLE_SPIDER.PY

```
1  import scrapy
2
3
4  class QuotesSpider(scrapy.Spider):
5      name = "quotes"
6      start_urls = [
7          'http://quotes.toscrape.com/tag/humor/',
8      ]
9
10     def parse(self, response):
11         for quote in response.css('div.quote'):
12             yield {
13                 'text': quote.css('span.text::text').extract_first(),
14                 'author': quote.xpath('span/small/text()').extract_first(),
15             }
16
17         next_page = response.css('li.next a::attr("href")').extract_first()
18         if next_page is not None:
19             next_page = response.urljoin(next_page)
20             yield scrapy.Request(next_page, callback=self.parse)
```

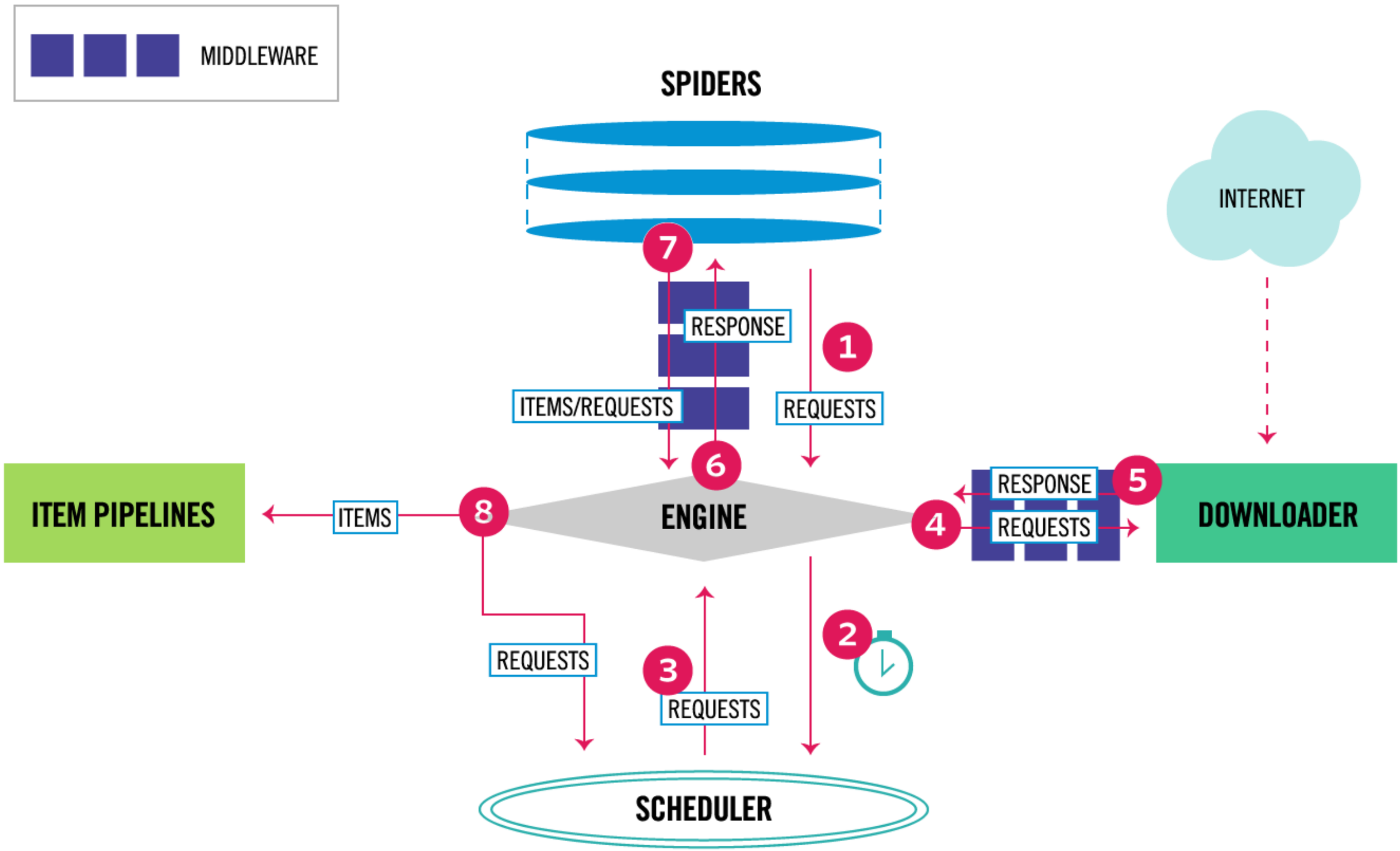
EXAMPLE_SPIDER.PY

Try different formats

```
$> scrapy runspider example_spider.py -o quotes.json
```


```
[
    {
        "author": "Jane Austen",
        "text": "\u201cThe person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.\u201d"
    },
    {
        "author": "Steve Martin",
        "text": "\u201cA day without sunshine is like, you know, night.\u201d"
    },
    {
        "author": "Garrison Keillor",
        "text": "\u201cAnyone who thinks sitting in church can make you a Christian must also think that sitting in a garage can make you a car.\u201d"
    },
    ...
]
```

АРХИТЕКТУРА SCRAPY



СОЗДАЕМ ПРОЕКТ

```
$> docker-compose run demo bash
$> scrapy startproject auto
```

СТРУКТУРА

созданного проекта

ГЕНЕРИРУЕМ SPIDER

```
$> cd auto && scrapy genspider autoru auto.ru
```

SPIDER STEP#0

```python
import scrapy


class AutoruSpider(scrapy.Spider):
    name = "autoru"
    allowed_domains = ["auto.ru"]
    start_urls = ['http://auto.ru/']

    def parse(self, response):
        pass
```

```
$> scrapy crawl autoru
```

SPIDER STEP# I

```
1  import scrapy
2
3
4  class AutoruSpider(scrapy.Spider):
5      name = 'autoru'
6
7      def start_requests(self):
8          url = 'https://auto.ru/cars/nissan/juke/i/group-offroad_5d/mod-77026/
statistics/'
9          yield scrapy.Request(url=url, callback=self.parse)
10
11     def parse(self, response):
12         filename = 'nissan-juke.html'
13
14         with open(filename, 'wb') as f:
15             f.write(response.body)
16
17         self.log('Saved file %s' % filename)
```

```
$> scrapy crawl autoru
```

SPIDER PYTHON SHELL

```
$> scrapy shell 'https://auto.ru/cars/nissan/juke/i/group-offroad_5d/
mod-77026/statistics/'

[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x7fd49dfa59b0>
[s]   item       {}
[s]   request    <GET https://auto.ru/cars/nissan/juke/i/group-.../>
[s]   response   <200 https://auto.ru/cars/nissan/juke/i/group-.../>
[s]   settings   <scrapy.settings.Settings object at 0x7fd49dfa5b00>
[s]   spider     <DefaultSpider 'default' at 0x7fd49dd19f60>
[s] Useful shortcuts:
[s]   shelp()           Shell help (print this help)
[s]   fetch(req_or_url) Fetch request (or URL) and update local objects
[s]   view(response)    View response in a browser
```

SPIDER PYTHON SHELL

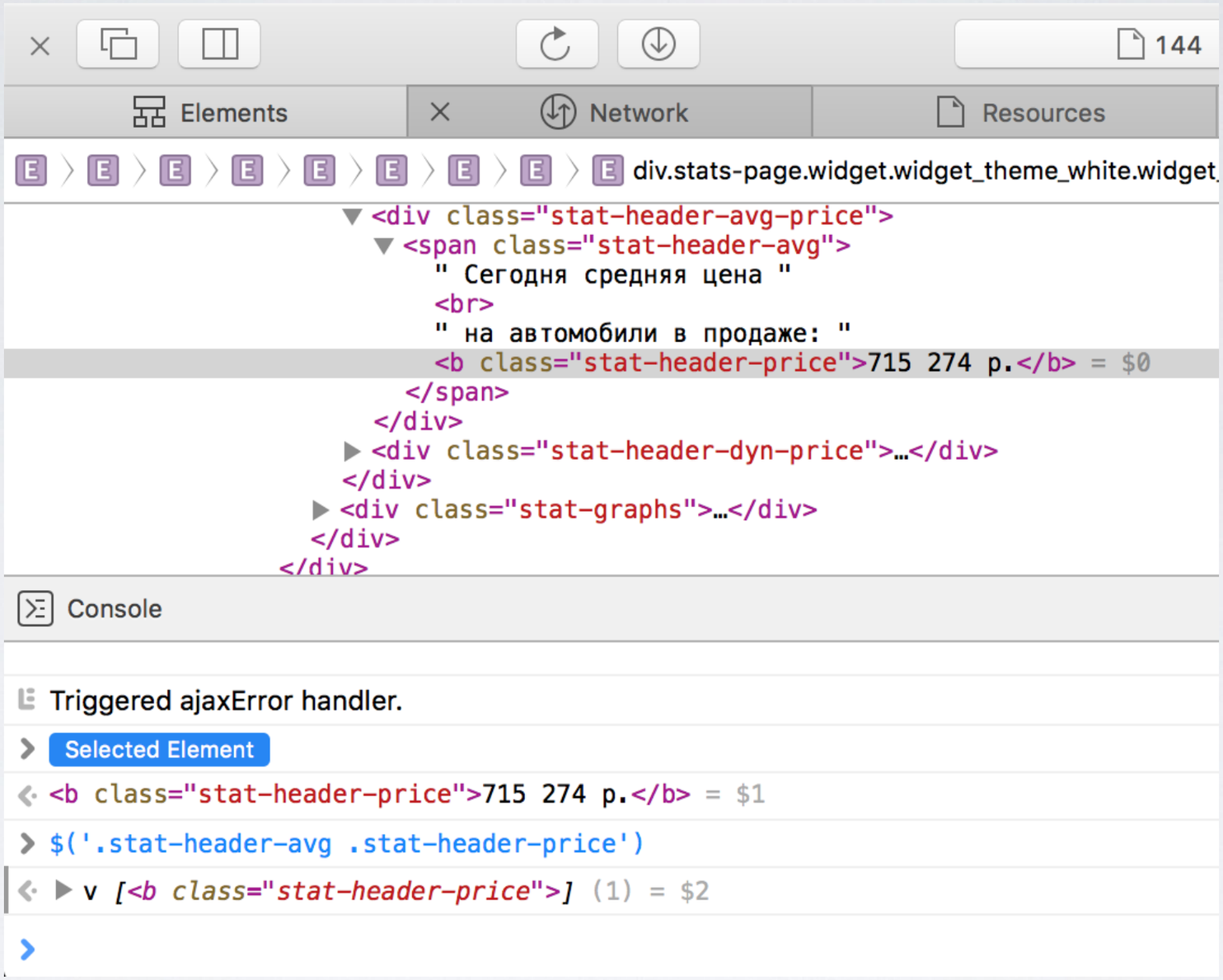
```
>>> response.css('title')
[<Selector xpath='descendant-or-self::title' data='<title>Продажа транспортных
средств</tit'>]

>>> response.css('title').extract()
['<title>Продажа транспортных средств</title>']

>>> response.css('title::text').extract()
['Продажа транспортных средств']

>>> response.css('title::text').extract_first()
'Продажа транспортных средств'
```

SPIDER PYTHON SHELL




```
>>> response.css('.stat-header-avg .stat-header-price::text').extract_first()
'715 274 р.'
```

SPIDER STEP#2

```
1  import scrapy
2
3
4  class AutoruSpider(scrapy.Spider):
5      name = 'autoru'
6
7      def start_requests(self):
8          url = 'https://auto.ru/cars/nissan/juke/i/group-offroad_5d/mod-77026/
statistics/'
9          yield scrapy.Request(url=url, callback=self.parse)
10
11     def parse(self, response):
12         raw_price = response.css('.stat-header-avg .stat-header-
price::text').extract_first()
13         print(raw_price)
```

```
$> scrapy crawl autoru
```

SPIDER STEP#3

```
1  import scrapy
2  from auto import items
3
4
5  class AutoruSpider(scrapy.Spider):
6      name = 'autoru'
7
8      def start_requests(self):
9          url = 'https://auto.ru/cars/nissan/juke/i/group-offroad_5d/mod-77026/
statistics/'
10         yield scrapy.Request(url=url, callback=self.parse)
11
12     def parse(self, response):
13         price = response.css('.stat-header-avg .stat-header-
price::text').extract_first()
14
15         if price is not None:
16             price = ''.join(price.split()[:-1])
17             yield items.AutoItem(brand='Nissan', model='Juke', price=price)
18         else:
19             yield items.AutoItem(error='Unable to find the price.')
```

```
$> scrapy crawl autoru
```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

auto.yandex.ru

https://auto.yandex.ru/stats/toyota/camry?body_type=SEDAN&year_from=2014&year_to=2016&generation_id=20229987

DB && PIPELINES

Peewee is a simple and small ORM

DEPLOY && SHEDULING

- Scrapyd
- Scrapy-json-rpc
- Scrapinghub

http -v POST localhost:6080/crawler/spiders jsonrpc=2.0 method=list param:=\[\] id:=1

AVOIDING GETTING BANNED

- rotate your user agent from a pool of well-known ones from browsers (google around to get a list of them)
- disable cookies (see COOKIES_ENABLED) as some sites may use cookies to spot bot behaviour
- use download delays (2 or higher). See DOWNLOAD_DELAY setting.
- if possible, use Google cache to fetch pages, instead of hitting the sites directly
- use a pool of rotating IPs. For example, the free Tor project or paid services like ProxyMesh. An open source alterantive is scrapoxy, a super proxy that you can attach your own proxies to.
- use a highly distributed downloader that circumvents bans internally, so you can just focus on parsing clean pages. One example of such downloaders is Crawlera

LINKS

- Scrapy doc.scrapy.org/en/latest/
- Scrapy doc.scrapy.org/en/latest/topics/practices.html
- Scrapy github.com/feiskyer/scrapy-examples
- Scrapyd scrapyd.readthedocs.io/en/stable/
- Scrapy JSON RPC github.com/scrapy-plugins/scrapy-jsonrpc
- Scrapinghub scrapinghub.com
- Peewee docs.peewee-orm.com/en/latest/
- Python diveintopython3.net
- Python learncodethehardway.org/python/

СПАСИБО ЗА ВНИМАНИЕ!