venv

Create venv

```
python3 -m venv tutorial-env
```

activate it

```
source tutorial-env/bin/activate
```

installing, upgrading, uninstalling and checking packages with pip

```
(tutorial-env) $ pip install novas
(tutorial-env) $ pip install requests==2.6.0
(tutorial-env) $ pip install --upgrade requests
(tutorial-env) $ pip uninstall novas
(tutorial-env) $ pip show requests
```

list of installed packages

```
(tutorial-env) $ pip list
```

save list of application required packages

```
(tutorial-env) $ pip freeze > requirements.txt
```

install list of required packages

```
(tutorial-env) $ pip install -r requirements.txt
```

create a project

```
scrapy startproject tutorial
```

create spider in spiders directory

```
$ scrapy genspider quotes example.com
Created spider ‘quotes' using template 'basic'
```

Then run spider with

```
scrapy crawl quotes
```

scrapy shell

```
scrapy shell 'http://quotes.toscrape.com/page/1/'
```

extract text from tag

```
>>> response.css('title::text').getall()
['Quotes to Scrape']
>>> response.css('title::text').get()
'Quotes to Scrape'
>>> response.css('title::text')[0].get()
'Quotes to Scrape'
>>> response.css('title::text').re(r'Quotes.*')
['Quotes to Scrape']
>>> response.css('title::text').re(r'Q\w+')
['Quotes']
>>> response.css('title::text').re(r'(\w+) to (\w+)')
['Quotes', 'Scrape']
>>> response.xpath('//title')
[<Selector xpath='//title' data='<title>Quotes to Scrape</title>'>]
>>> response.xpath('//title/text()').get()
'Quotes to Scrape'
>>> response.css('a[href*=image] img::attr(src)').getall()
['image1_thumb.jpg',
 'image2_thumb.jpg',
 'image3_thumb.jpg',
 'image4_thumb.jpg',
 'image5_thumb.jpg’]
```

save scraped data to JSON once

```
scrapy crawl quotes -o quotes.json
```

several times

```
scrapy crawl quotes -o quotes.jl
```

get href of a link

```
>>> response.css('li.next a::attr(href)').get()
'/page/2/'
>>> response.css('li.next a').attrib['href']
'/page/2/'
```

FULL SPIDER

```
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
    ]

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').get(),
                'author': quote.css('small.author::text').get(),
                'tags': quote.css('div.tags a.tag::text').getall(),
            }

        next_page = response.css('li.next a::attr(href)').get()
        if next_page is not None:
            next_page = response.urljoin(next_page)
            yield scrapy.Request(next_page, callback=self.parse)
```

OR

```
        if next_page is not None:
            yield response.follow(next_page, callback=self.parse)
```

CSS selector in loop

```
for href in response.css('ul.pager a::attr(href)'):
    yield response.follow(href, callback=self.parse)
```

for URLs «a» it can be just

```
for a in response.css('ul.pager a'):
    yield response.follow(a, callback=self.parse)
```

To create multiple requests from an iterable

```
anchors = response.css('ul.pager a')
yield from response.follow_all(anchors, callback=self.parse)
```

OR

```
yield from response.follow_all(css='ul.pager a', callback=self.parse)
```

Scraping all author info

```
import scrapy



class AuthorSpider(scrapy.Spider):
    name = 'author'

    start_urls = ['http://quotes.toscrape.com/']

    def parse(self, response):
        author_page_links = response.css('.author + a')
        yield from response.follow_all(author_page_links, self.parse_author)

        pagination_links = response.css('li.next a')
        yield from response.follow_all(pagination_links, self.parse)

    def parse_author(self, response):
        def extract_with_css(query):
            return response.css(query).get(default='').strip()
```

```
        yield {
            'name': extract_with_css('h3.author-title::text'),
            'birthdate': extract_with_css('.author-born-date::text'),
            'bio': extract_with_css('.author-description::text'),
        }
```

Using arguments (as variables)

```
scrapy crawl quotes -o quotes-humor.json -a tag=humor
```

```
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        url = 'http://quotes.toscrape.com/'
        tag = getattr(self, 'tag', None)
        if tag is not None:
            url = url + 'tag/' + tag
        yield scrapy.Request(url, self.parse)

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').get(),
                'author': quote.css('small.author::text').get(),
            }

        next_page = response.css('li.next a::attr(href)').get()
        if next_page is not None:
            yield response.follow(next_page, self.parse)
```