پروژه beautiful Soup برای استخراج داده از سایت http://quotes.toscrape.com/

قصد داریم از quoteهای داخل صفحه، متن هر quote و author آن و tag های زیر آن را استخراج کنیم

برای استفاده از beautiful soup ابتدا آن را ایمپورت میکنیم به شکل زیر :

```
from bs4 import BeautifulSoup
```

قبل از آن باید ماژول bs4 را به وسیله دستور pip (یا هر روش مناسب دیگری) نصب کرده باشیم

ماژول requests هم به منظور ایجاد درخواست ها باید ایمپورت شود :

```
import requests
```

همچنین به ماژول csv نیز به منظور ایجاد فایل های csv احتیاج داریم

```
from csv import writer
```

از آنجا که در این پروژه فقط میخواهیم نتیجه کد را در یک فایل خروجی بنویسیم، ایمپورت کردن writer به تنهایی از این ماژول کفایت میکند

با استفاده از متد .get() از ماژول requests کانتنت صفحه مورد نظر را دریافت کرده و درون متغیر response میریزیم

```
response = requests.get("http://quotes.toscrape.com")
```

حالا با استفاده از تابع BeautifulSoup یک اینستنس از متن response که به دست آورده بودیم را میسازیم و با تجزیه گر html.parser، آن را تجزیه میکنیم

در صورتیکه این پارامتر را تعیین نکنیم خطایی رخ نمیدهد بلکه فقط به ما هشدار میدهد که خودش متناسب با سیستم ما، یک تجزیه گر انتخاب میکند ولی ممکن است که این تجزیه گر روی سیستم یا سیستم های دیگر متفاوت باشد، بهتر است که آن را مشخص کنیم

```
soup = BeautifulSoup(response.text, "html.parser")
```

حالا آماده آغاز استخراج هستیم

با استفاده از امکان insepct element مرورگر، درمیابیم که کانتنت های مورد نظر ما در تگ های divایی قرار دارند که کلاس quote دارند

با استفاده از متد find_all(). همه آنها را در لیستی به اسم quotes ذخیره میکنیم

```
quotes = soup.find_all("div", class_="quote")
```

خروجی متد find_all(). یک لیست است

بعد یک فایل را به منظور نوشتن (یعنی با مد w) باز میکنیم و اسم csv_file را برای آن قرار میدهیم، قرار است که درون این فایل نتیجه کار را ببینیم

```
with open("quotes_data.csv", 'w') as csv_file:
```

برای نوشتن در فایل توسط ماژول csv از تابع writer استفاده میکنیم

```
csv_writer = writer(csv_file)
```

می خواهیم header را ایجاد کنیم که حاوی رشته های Text و Author و Tags است، برای این کار از متد writerow(). استفاده میکنیم، پارامتر این متد یک لیست است که موارد مورد نظر ما داخل آن قرار میگیرند

```
csv_writer.writerow(["Text", "Author", "Tags"])
```

حالا باید تک تک quoteها را پیمایش کنیم و text و author و tags هر کدام را استخراج کنیم، پس در لیست quotes که قبلاً ایجاد کرده بودیم پیمایش میکنیم

```
for quote in quotes:
```

برای هر quote اقدامات زیر را انجام میدهیم

میدانیم که متن هر quote داخل المنتی است به اسم span، به وسیله متد find(). آن تگ را پیدا میکنیم و به وسیله متد get_text(). به متن آن دسترسی پیدا میکنیم، و حاصل را درون متغیری به اسم text ذخیره میکنیم

```
text = quote.find("span").get_text()
```

بعد نوبت به author میرسد که مانند قبلی است با این تفاوت که باید در تگ small به دنبال آن بگردیم

```
author = quote.find("small").get_text()
```

بعد نوبت به tags میرسد، اینجا یک مقدار کار پیچیده میشود، تگ هایی که به دنبالشان هستیم در لینک ها (تگ های a)یی قرار دارند که آن لینکها خودشان در divهایی با کلاس tags قرار دارند، پس ابتدا aهایی که در divهایی با کلاس tags قرار دارند را با استفاده از متد find_all(). پیدا میکنیم و درون متغیر a_tags میریزیم

```
a_tags = quote.find("div", class_="tags").find_all("a")
```

بعد یک لیست تهی به اسم tags ایجاد میکنیم و با یک حلقه for لیست a_tags را پیمایش کرده و متن تک تک آنها را به آن لیست append میکنیم

```
for a_tag in a_tags:
    tags.append(a_tag.get_text())
```

در نهایت هم متغیرهای text و author و tags را با استفاده از متد writerow(). به هر سطر اختصاص میدهیم

```
csv_writer.writerow([text, author, tags])
```

بعد از اجرا گرفتن از برنامه، یک فایل با پسوند csv ایجاد میشود که حاوی اطلاعات متن و نویسنده و تگها برای هر quote از سایت http://quotes.toscrape.com است

پروژه Scrapy برای استخراج داده از سایت http://quotes.toscrape.com/

قصد داریم از quoteهای داخل صفحه، متن هر quote و author آن و tag های زیر آن را استخراج کنیم

برای کار با اسکرپی ماژول های Lxml و Parsel و W3lib و Twisted و Cryptography و pyopenssl باید از قبل نصب شده باشند

Lxml برای تجزیه کردن XML و HTML است

Parsel یک کتابخانه اکسترکتور HTML/XML است علاوه بر lxml

W3lib یک هلپر همه منظوره برای کار کردن با URL ها و انکودینگ وب پیج ها است

Twisted یک فریم ورک شبکه نامتقارن یا غیر همزمان است

Cryptography و pyopenssl برای هندل کردن نیازهای مختلف امنیتی هستند

به دلیل اینکه من از anaconda استفاده میکنم همه ماژول های مورد نظر از قبل نصب شده بودند و مشکلی برای اجرای کد به وجود نیامد

برای برپایی پروژه ابتدا با استفاده از دستور کامندی زیر پوشه ایی که میخواهیم پروژه مان را درونش ایجاد کنیم میسازیم

```
scrapy startproject tutorial
```

با زدن این دستور در anaconda command line یک پوشه به اسم tutorial ساخته میشود که حاوی یک سری فایلها و فولدرهای آماده به منظور کار با Scrapy است، به پوشه داخلی spiders وارد میشویم و پروژه را در آن ایجاد میکنیم

اسکرپی را ایمپورت میکنیم

```
import scrapy
```

میبینیم که برخلاف beautiful soup دیگر نیازی به ماژول های requests و csv نداریم و خود scrapy این نیازها را برای ما برآورده میکند

برای ایجاد یک اسپایدر باید کلاس بسازیم، این کلاس باید از scrapy.spider ارث بری کند

```
class QuotesSpider(scrapy.Spider):
```

بعد یک اسم برای آن میگذاریم و داخل اتریبیوت name ذخیره میکنیم، این اتریبیوت حتماً باید name باشد و چیز دیگری نمیتواند باشد

```
name = "quotes"
```

بعد یک لیست از url یا urlهایی که میخوایم به آنها درخواست ارسال کنیم را در اتریبیوت start_urls ذخیره کنیم، این لیست هم باید با همین نام باشد چراکه scrapy برای ایجاد درخواست ها انتظار این اسم را میکشد

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

بعد متد parse را تعریف میکنیم، این متد هم باید همین اسم را داشته باشه، scrapy انتظار دارد که برای هر اسپایدری که میسازیم این متد را تعریف کنیم، پارامتر response هم برای آن ثابت است، response در واقع به هر پاسخی که از درخواست http دریافت میکنیم اشاره میکند

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

به همین راحتی آماده سازی برای استخراج داده انجام میشود

از آنجا که میدانیم روی صفحه به دنبال divهایی از کلاس quoteهستیم، با یک حلقه for همه آنها را پیمایش میکنیم

```
for quote in response.css('div.quote'):
```

بعد با yield در هر مرحله یک دیکشنری ایجاد میکنیم که دارای کلیدهای text و author و tags و مقادیر مرتبط با هرکدام است

برای متن هر quote با استفاده از متد ()css. متن spanهایی از کلاس text را با استفاده از متد ()get. داخل کلید text ذخیره میکنیم

```
'text': quote.css('span.text::text').get()
```

بعد برای نویسنده هر quote مثل مورد قبلی عمل میکنیم با این تفاوت که این بار smallهایی از کلاس author را در کلید author ذخیره میکنیم

```
'author': quote.css('small.author::text').extract_first()
```

()get. و ()extract_first. در خروجی فرق چندانی با هم نمیکنند و فقط خواستم از هردوی آنها استفاده کرده باشم

و در نهایت هم متن لینکها (a ها)یی که از کلاس tag که داخل divهایی از کلاس tags هستند را در کلید tags میریزیم، به این دلیل که این تگ ها در هر quote ممکن است از یکی بیشتر باشند، از متد ()getall برای دریافت آنها استفاده میکنیم که خروجی آن یک لیست است

```
'tags': quote.css('div.tags a.tag::text').getall()
```

میبینیم که برای این مورد آخر چقدر کارمان نسبت به استفاده از beautiful soup راحت تر شد

کار استخراج داده تمام شد حالا با دستور زیر یک بار اسپایدر را اجرا میکنیم

```
scrapy crawl quotes
```

و بعد برای خروجی گرفتن و ذخیره در یک فایل csv از دستور زیر استفاده میکنیم

```
scrapy crawl quotes -O quotes.csv
```

در صورتی که بخواهیم در خروجی یک فایل json یا xml ایجاد کنیم هم میتوانیم به راحتی در اخر این دستور، خروجی مورد نظر را مشخص کنیم

برای json :

```
scrapy crawl quotes -O quotes.json
```

برای xml :

scrapy crawl quotes -O quotes.xml

به همین راحتی یک فایل با پسوند مورد نظر ما ایجاد میشود که حاوی اطلاعات متن و نویسنده و تگها برای هر quote از سایت http://quotes.toscrape.com است

مقایسه beautiful soup و scrapy

مزیت های scrapy نسبت به beautiful soup

برای کار با scrapy نیاز به ایمپورت کردن و کار با ماژول های اضافی نیست

برای دسترسی به تگهای داخلی با کلاسهای مختلف، توسط Scrapy و تنها در یک خط قادر به انجام این کار هستیم، که این مورد در beautiful soup پیچیده میشود

برای ایجاد فایل خروجی توسط scrapy به راحت ترین شکل ممکن و تنها با یک کامند قادر به انجام آن خواهیم بود، در حالیکه برای beautiful soup مجبور به باز کرن فایل و همچنین استفاده از ماژول csv هستیم که در نهایت هم فقط یک نوع خروجی به ما میدهد و آن هم csv است

مزیت beautiful soup نسبت به scrapy

به عنوان تنها مزیت beautiful soup میتوان به این موضوع اشاره کرد که flexibility بیشتری دارد، مانند اسمهای متغیرها و متدها که در scrapy حتما باید همان هایی را بگذاریم که scrapy انتظار دارد

به هر حال اگر مشکلی از این بابت ندارید که flexibility را فدای سرعت و ساده تر کد زدن کنید، از scrapy استفاده کنید

استاندارد کردن کدها طبق pep8

استفاده از pyest testing framework

باید ابتدا pytest را نصب کنیم بعد pep8 را به آن اضافه کنیم

Pip install pytest

Pip install pytest-pep8

البته این ماژول در طول زمان تغییر نام داده و تبدیل به pycodestyle شده است، پس با دستور زیر میتوان آن را نصب کرد

```
Pip install pycodestyle
```

البته اگر از Anaconda استفاده میکنید همه آنها از قبل نصب شده و آماده هستند

برای استفاده از آن در cmd وارد مسیر برنامه میشویم، و دستور زیر را وارد میکنیم

```
Pycodestyle Crawler.py
```

یعنی اول دستور pycodestyle و بعد نام فایل پایتون مورد نظر

در نتیجه اجرای دستور بالا یک گزارش برای ما نمایش داده میشود که حاوی اشکالات موجود در کد و همچنین اقداماتی است که انتظار دارد برای رفع آن اشکالات انجام بدهیم، با برطرف کردن تک تک آنها به استایل مورد نظر دست میابیم