上传到群里的资源文件：

- python-3.9.0-amd64.exe (windows python)
- python-3.9.0-macosx10.9.pkg(mac)
- pycharm-community-2020.2.4.exe (windows下pycharm的社区版)
- pycharm-community-2020.2.4.dmg(mac下pycharm的社区版)
- python和爬虫.pdf（本文件）
- python和爬虫.zip(演示文件，jupyter)
- requirements.txt

下面是大纲，具体的演示文件稍后会发。

下面从以下几个部分开始介绍：

- Python环境和IDE
- Python语法和特性简介
- 爬虫

Python环境和IDE

- python 开发环境：https://www.python.org/

记得勾选添加环境变量。
如果切换地址，最好**不要**放到C盘除用户文件夹（即`C:\Users\{用户名}`）的目录下。

安装第三方库

Win+R输入`cmd`。打开cmd，执行下面步骤。（mac打开shell）

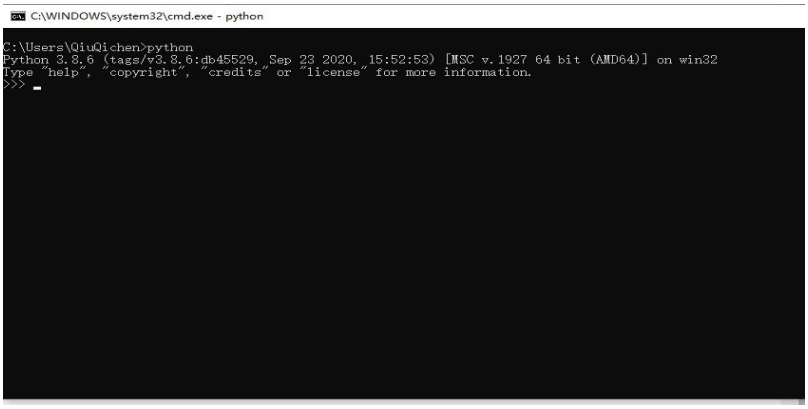
- (**可选步骤**)永久切换pip的源:
  `pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple`
- 安装
  `pip install -r requirements.txt -i  https://pypi.tuna.tsinghua.edu.cn/simple --user`
  注意requirements.txt需要在当前cmd或者shell的目录下。

- 如果教程结束后想删掉这些库可以通过：
  `pip uninstall -r requirement.txt`
  或者逐一删除。

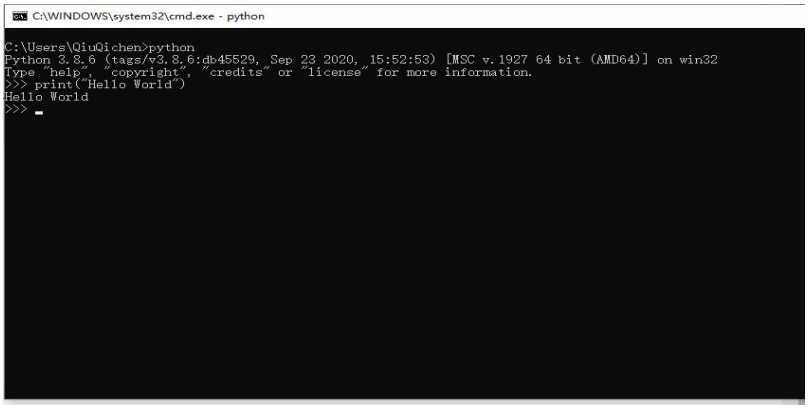
- requirement.txt里的第三方库有:
  - jupyter notebook（如果有npm，也可以装jupyter lab）
  - ipython（更好用的python交互界面）
  - requests（http）
  - bs4（解析html）
  - lxml（xpath解析）
  - scrapy（高性能的爬虫框架）
  - tqdm(进度条)

使用python交互界面

推荐用ipython REPL
在命令行下，输入`python`



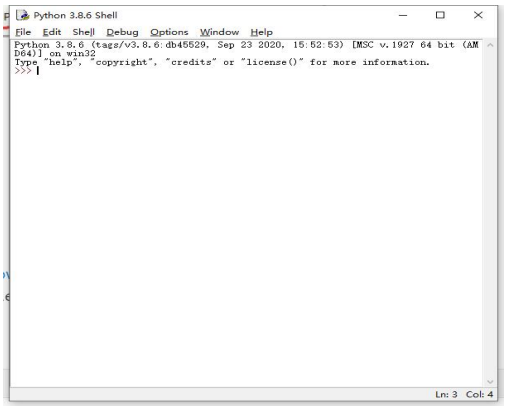
然后，输入`print('Hello world!')`



退出界面的方法是按 Ctrl+Z 或者执行 exit()。

常见的IDE

IDLE



自带的编辑器，不推荐。

Pycharm

https://www.jetbrains.com/pycharm/download/#section=windows
可以社区版(community)也可用专业版(professional)（使用@bupt.edu.cn邮箱免费激活）
pycharm的安装包已分享在群里。

Vscode

安装python插件，修改launch.json或者用Code Runner

jupyter notebook(或者jupter lab)

命令行下：

```
pip install jupyter notebook -i https://pypi.tuna.tsinghua.edu.cn/simple
```

启动时，cmd内输入：
`jupyter notebook`

Python语法的简单介绍

基本说明

格式、注释和换行

格式

python缩进要求是每层使用4个空格

缩进一般用TAB键（在大小写键的上面）。一般Tab键都会被编辑器替换成4个空格。

注释

是单行注释
""" """可以作为多行注释

跨行

代码后面加上\可以跨行。
如果有逗号，不用\也可

输出

print的简单使用

一些常见的内置函数

- id
- type
- isinstance
- issubclass

如何获取方法、类的用法?

- help
- ipyhton ?
- ipython ??
- dir

数字类型和操作符

- int
- 虚数
- **
- //
- `and` `or` `not`
- 三目表达式

流程控制

- if else
- for和while
- break和continue
- try
- else
- pass
- with
- 函数

常用容器

- list
- 列表的索引
- str
- dict

类的简单介绍

- __init__方法
- magic method

- 继承

库

pip

`pip install ...`
`pip uninstall ...`
`pip list|grep scrapy` (liunx,mac)
`pip list| findstr /i scrapy` (windows)

入门教程参考

- 官方入门教程
- microsoft ai-edu的python教程
- 廖雪峰python教程
- (书)python编程从入门到实践

爬虫

本节将介绍

- 爬虫的概念
- HTTP
- 爬虫的架构
- requests
- 数据的解析
- 数据的存储
- 例子
- scrapy

概念介绍

爬虫是什么

> 网络爬虫（英语：web crawler），也叫网络蜘蛛（spider），是一种用来自动浏览万维网的网络机器人。其目的一般为编纂网络索引。

> 网络爬虫（又称为网页蜘蛛，网络机器人），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本

爬虫的道德和法律问题

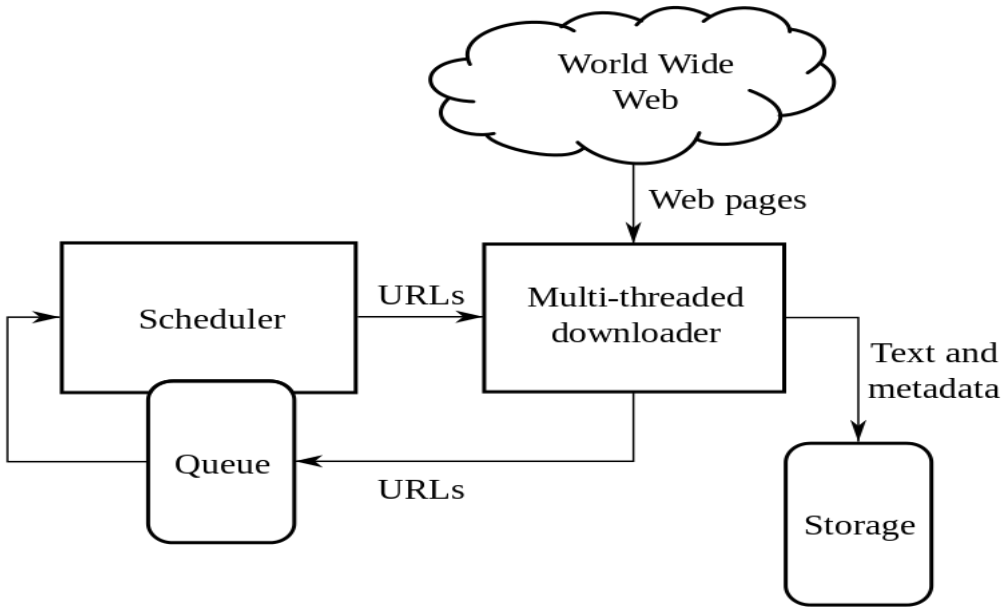
- robots.txt

> robots.txt（统一小写）是一种存放于网站根目录下的ASCII编码的文本文件，它通常告诉网络搜索引擎的漫游器（又称网络蜘蛛），此网站中的哪些内容是不应被搜索引擎的漫游器获取的，哪些是可以被漫游器获取的。
>
> ...
> 这个协议也不是一个规范，而只是约定俗成的.

- 爬虫的法律问题
    - 网络爬虫不能攻击服务器（访问频率不能太高）
    - 爬虫不能涉及个人隐私数据抓取与贩卖

    - 爬虫不能利用无版权的商业数据获利
      参考：
- [爬虫究竟是合法还是违法的？](#)
- [中国爬虫违法违规案例汇总](#)

爬虫的结构

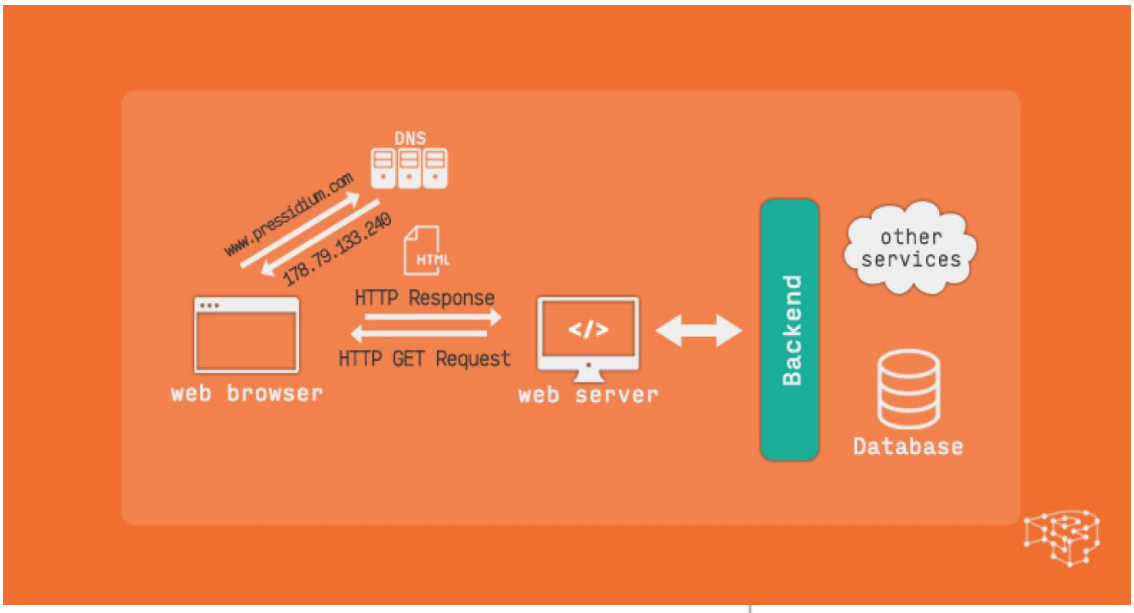


HTTP

参考：[渲染页面：浏览器的工作原理](#)

- URL
    - 统一资源定位器
    - 协议、主机、端口、路径

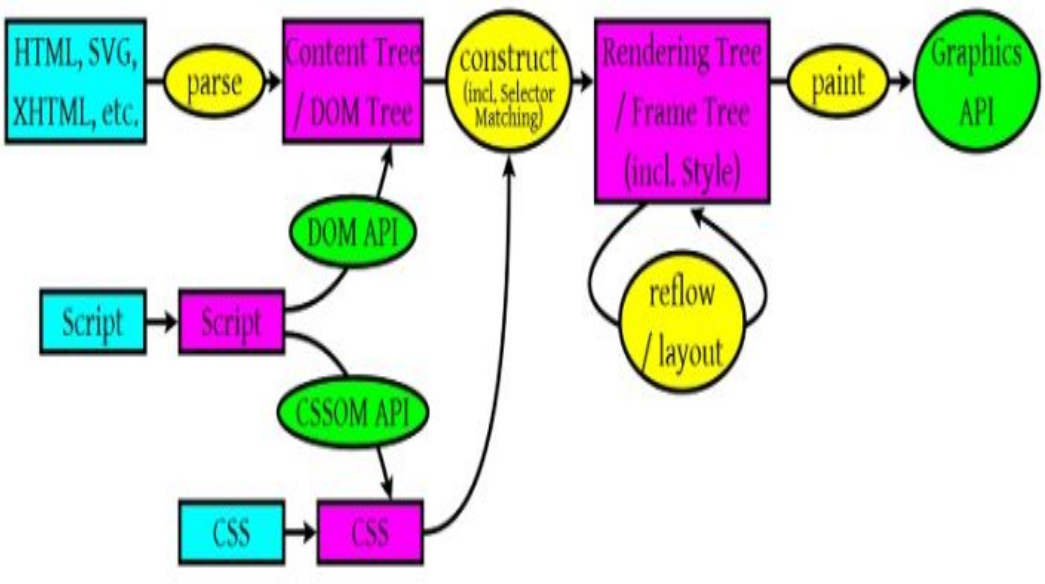
- 如何获取到资源



> 基本上，当数据在Web上传输时，是以成千上万的小数据块的形式传输的。大量不同的用户都可以同时下载同一个网页。如果网页以单个大的数据块形式传输，一次就只有一个用户下载，无疑会让Web非常没有效率并且失去很多乐趣。

- 网页如何解析、渲染
  - 网页可以分为三大部分一 - HTML , CSS 和 JavaScript

- 渲染的过程



- HTTP请求和抓包
  - Chrome dev tools
  - HTTP包的内容

❑ Accept：请求报头域，用于指定客户端可接受哪些类型的信息。
❑ Accept-Language：指定客户端可接受的语言类型。
❑ Accept-Encoding：指定客户端可接受的内容编码。
❑ Host：用于指定请求资源的主机 IP 和端口号，其内容为请求 URL 的原始服务器或网关的位置。从 HTTP 1.1 版本开始，请求必须包含此内容。
❑ Cookie：也常用复数形式 Cookies，这是网站为了辨别用户进行会话跟踪而存储在用户本地的数据。它的主要功能是维持当前访问会话。例如，我们输入用户名和密码成功登录某个网站后，服务器会用会话保存登录状态信息，后面我们每次刷新或请求该站点的其他页面时，会发现都是登录状态，这就是 Cookies 的功劳。Cookies 里有信息标识了我们所对应的服务器的会话，每次浏览器在请求该站点的页面时，都会在请求头中加上 Cookies 并将其发送给服务器，服务器通过 Cookies 识别出是我们自己，并且查出当前状态是登录状态，所以返回结果就是登录之后才能看到的网页内容。
❑ Referer：此内容用来标识这个请求是从哪个页面发过来的，服务器可以拿到这一信息并做相应的处理，如做来源统计、防盗链处理等。
❑ User-Agent：简称 UA，它是一个特殊的字符串头，可以使服务器识别客户使用的操作系统及版本、浏览器及版本等信息。在做爬虫时加上此信息，可以伪装为浏览器；如果不加，很可能会被识别出为爬虫。
❑ Content-Type：也叫互联网媒体类型（Internet Media Type）或者 MIME 类型，在 HTTP 协议消息头中，它用来表示具体请求中的媒体类型信息。例如，text/html 代表 HTML 格式，image/gif 代表 GIF 图片，application/json 代表 JSON 类型，更多对应关系可以查看此对照表：http://tool.oschina.net/commons

requests

- requests.get()
- response.json()
- 自定义header
- 传递url参数
- 设置超时
- 下载图片

几个实例

北邮官网

编码问题

```
requests.get("https://www.bupt.edu.cn/",headers=headers)
```

百度

header
`requests.get("https://www.baidu.com")`

数据的解析

- bs4
- lxml

数据的存储

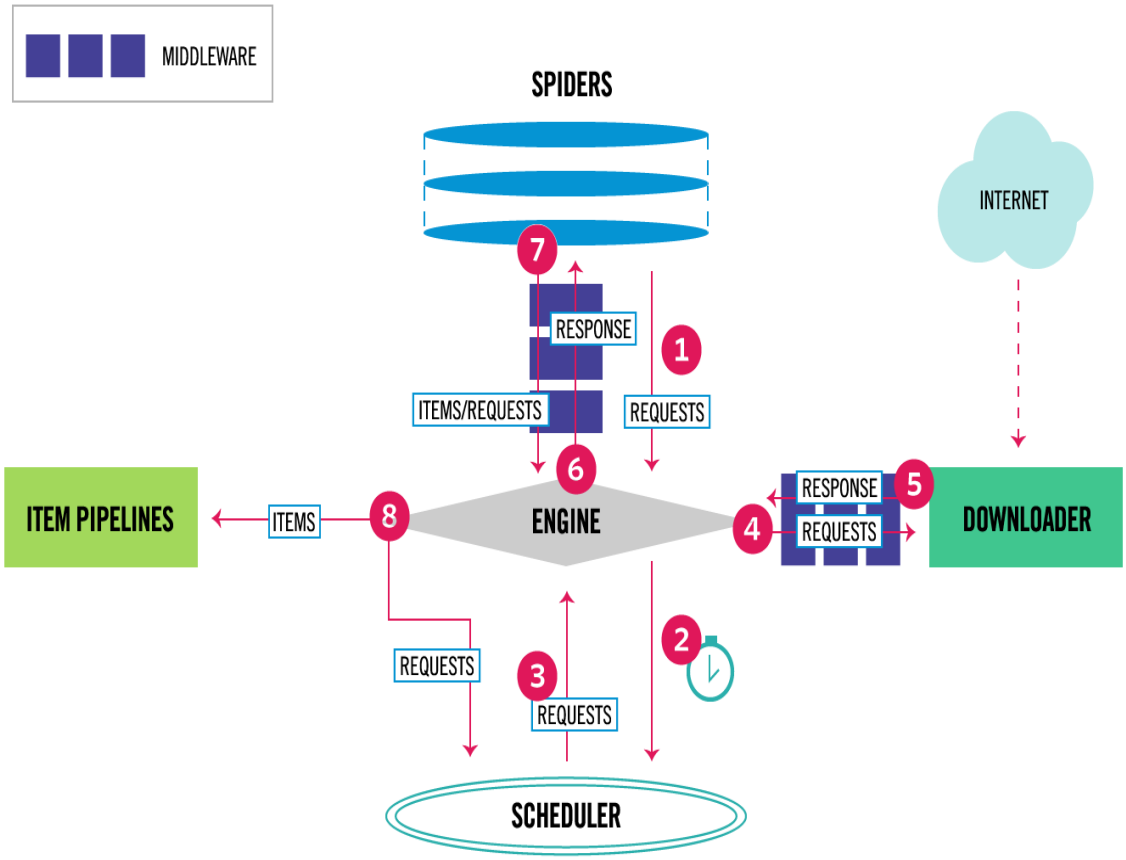
- 文本格式
- csv
- json
- pickle

知乎api的爬虫

- `API： https://www.zhihu.com/api/v4/questions/{问题id}/answers?include=data%5B%2A%5D.is_normal%2Cadmin_closed_comment%2Creward_info%2Cis_collapsed%2Cannotation_action%2Cannotation_detail%2Ccollapse_reason%2Cis_sticky%2Ccollapsed_by%2Csuggest_edit%2Ccomment_count%2Ccan_comment%2Ccontent%2Ceditable_content%2Cvoteup_count%2Creshipment_settings%2Ccomment_permission%2Ccreated_time%2Cupdated_time%2Creview_info%2Crelevant_info%2Cquestion%2Cexcerpt%2Crelationship.is_authorized%2Cis_author%2Cvoting%2Cis_thanked%2Cis_nothelp%2Cis_labeled%2Cis_recognized%2Cpaid_info%2Cpaid_info_content%3Bdata%5B%2A%5D.mark_infos%5B%2A%5D.url%3Bdata%5B%2A%5D.author.follower_count%2Cbadge%5B%2A%5D.topics%3Bsettings.table_of_content.enabled&limit={每次条数}&offset={偏置}&platform=desktop&sort_by=default`

scrapy的简单介绍

https://docs.scrapy.org/en/latest/



创建项目

`scrapy startporject [项目名称]`

例如，`scrapy startproject zhihu`

生成的文件结构如下：

```
zhihu/
    scrapy.cfg            # deploy configuration file

    zhihu/
        __init__.py

        items.py          # 定义对象

        middlewares.py    # 中间层

        pipelines.py      # 管道，用于保存信息

        settings.py       # 配置

        spiders/          # 网页抓取和解析部分，我们主要需要修改的地方
            __init__.py
```

产生一只爬虫

`scrapy genspider [名字] [domain]`

`scrapy genspider ans_spider zhihu.com`

配置的修改

```
COOKIES_ENABLED = True #打开cookies
DEFAULT_REQUEST_HEADERS = {
  'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36",
}# 默认User_agent
ROBOTSTXT_OBEY = False # 不遵守robots.txt
DOWNLOAD_DELAY = 0.5 #每次下载延时0.5s，可选
```

调试

`scrapy shell`

`scrapy shell [url]`

常用的参数

- `--nolog` 关闭log
- `-s USER_AGENT="<custom user agent>"` 添加自定义header

例如：

`scrapy shell -s USER_AGENT="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36" "https://www.zhihu.com/question/36132174" --nolog`

shell中的方法：

- view()
- fetch()
- response.xpath()
- response.css()
- get(),getall(),re()

更多调试方法：
https://docs.scrapy.org/en/latest/topics/shell.html
https://docs.scrapy.org/en/latest/topics/debug.html

编写爬虫文件

以知乎问题内容爬虫为例：

生成爬虫

`scrapy genspider ans_spider zhihu.com`

创建item

`Item`是装载数据的容器
在`./zhihu/items.py`下写入：

```
class AnswerItem(scrapy.Item):
    author = scrapy.Field()
    author_token = scrapy.Field()
    content =scrapy.Field()
    url = scrapy.Field()
```

编写爬虫和解析

在`./zhihu/spiders.py`下写入：

```
import scrapy
from ..items import AnswerItem
from scrapy import Request
class AnsSpider(scrapy.Spider):
    name = 'ans'
    allowed_domains = ['zhihu.com']
    api_url = "https://www.zhihu.com/api/v4/questions/{}/answers?
include=data%5B%2A%5D.is_normal%2Cadmin_closed_comment%2Creward_info%2Cis_collap
sed%2Cannotation_action%2Cannotation_detail%2Ccollapse_reason%2Cis_sticky%2Ccoll
apsed_by%2Csuggest_edit%2Ccomment_count%2Ccan_comment%2Ccontent%2Ceditable_conte
nt%2Cvoteup_count%2Creshipment_settings%2Ccomment_permission%2Ccreated_time%2Cup
dated_time%2Creview_info%2Crelevant_info%2Cquestion%2Cexcerpt%2Crelationship.is_
authorized%2Cis_author%2Cvoting%2Cis_thanked%2Cis_nothelp%2Cis_labeled%2Cis_reco
gnized%2Cpaid_info%2Cpaid_info_content%3Bdata%5B%2A%5D.mark_infos%5B%2A%5D.url%3
Bdata%5B%2A%5D.author.follower_count%2Cbadge%5B%2A%5D.topics%3Bsettings.table_of
_content.enabled&limit={}&offset={}&platform=desktop&sort_by=default"
    question_url = "https://www.zhihu.com/question/{}"

    def start_requests(self):
        for question_id in problem_id_list:
            yield Request(self.question_url.format(question_id),
callback=self.parse_question_page,
                          cb_kwargs={"question_id":question_id})
    def parse_question_page(self,response,question_id):
        nums = int(response.xpath(r'//*[@class="List-headerText"]/span/text()')
[0].extract())
        for start_from in range(0, nums, 20):
            yield Request(self.api_url.format(question_id, 20, start_from),
callback=self.parse_content,
                          cb_kwargs={"question_id":question_id})
    def parse_content(self, response,question_id):
        data = response.json()["data"]
        for ans in data:
            item = AnswerItem()
            item["author"] = ans["author"]["name"]
            item["author_token"] = ans["author"]["url_token"]
            item["content"] = ans["content"]
            item["url"] =
"".join((self.question_url.format(question_id),"/answer/",str(ans["id"])))
            yield item
```

启动爬虫

`scrapy crawl [爬虫名字]`

例如:
`scrapy zhihu ans`

添加命令行关键词参数

在`AnsSpider`中修改`start_requests`:

```
    def start_requests(self):
        ############################## 修改的部分
        problem_id_list = getattr(self, 'problem_id_list', [])
        #从参数列表
        problem_id_list = problem_id_list.split(',')
        ##############################

        for question_id in problem_id_list:
            yield Request(self.question_url.format(question_id),
callback=self.parse_question_page,
                          cb_kwargs={"question_id":question_id})
```

运行:
`scrapy crawl ans -a problem_id_list=36132174`

导出数据到csv

使用`-o`参数可以快捷的导出数据。
如果需要定制化的导出数据，可以参考：https://docs.scrapy.org/en/latest/topics/feed-exports.html

例子:
`scrapy crawl ans -a problem_id_list=36132174 -o data.csv`

pipline传图片

scrapy中的`image_pipline`可以用来传图片。（当然也可以自定义pipline）

- 首先需要自定义一个Item,里面含有`image_urls`和`images`两个field

```
class MyItem(scrapy.Item):
    # ... other item fields ...
    image_urls = scrapy.Field()
    images = scrapy.Field()
```

- 然后在`settings.py`中，添加（或者取消注释）：

```
ITEM_PIPELINES = {'scrapy.pipelines.images.ImagesPipeline': 1} #优先级为1
IMAGES_STORE = 'images' #保存目录地址
# IMAGES_URLS_FIELD = 'image_urls' # 可以修改图片url保存的field名字
# IMAGES_RESULT_FIELD = 'images' # 修改图片下载结果的field名字
# IMAGES_MIN_HEIGHT = 110 # 图片最小高度
# IMAGES_MIN_WIDTH = 110 # 图片最小宽度
```

- 最后在解析html里，把图片的url地址传入`image_urls`，然后yield 这个item即可。

在zhihu的项目文件中，有一个`img`的spider，可以用来爬取一个回答中所有的图片。

`scrapy crawl img -a problem_id_list=36132174`

传文件和图片的解决方案：https://docs.scrapy.org/en/latest/topics/media-pipeline.html

Image_pipline的文档：https://docs.scrapy.org/en/latest/topics/media-pipeline.html#scrapy.pipelines.images.ImagesPipeline

参考

- 《Python3网络爬虫开发实战》崔庆才
- A Web Crawler With asyncio Coroutines