# **Day01回顾**

## **请求模块(requests)**

html = requests.get(url=url,headers=headers).text  
html = requests.get(url=url,headers=headers).content.decode('utf-8')  
  
with open('xxx.txt','w',encoding='utf-8') as f:  
 f.write(html)

## **编码模块(urllib.parse)**

1、urlencode({dict})  
 urlencode({'wd':'美女','pn':'20'})  
 编码后 ：'wd=%E8%D5XXX&pn=20'  
  
2、quote(string)  
 quote('织女')  
 编码后 ：'%D3%F5XXX'  
  
3、unquote('%D3%F5XXX')

## **解析模块(re)**

* **使用流程**
* p = re.compile('正则表达式',re.S)  
  r\_list = p.findall(html)
* **贪婪匹配和非贪婪匹配**
* 贪婪匹配(默认) ： .\*  
  非贪婪匹配 ： .\*?
* **正则表达式分组**
* 【1】想要什么内容在正则表达式中加()  
  【2】多个分组,先按整体正则匹配,然后再提取()中数据。结果：[(),(),(),(),()]

## **抓取步骤**

【1】确定所抓取数据在响应中是否存在（右键 - 查看网页源码 - 搜索关键字）  
【2】数据存在: 查看URL地址规律  
【3】写正则表达式,来匹配数据  
【4】程序结构  
 a>每爬取1个页面后随机休眠一段时间

# 程序结构  
class xxxSpider(object):  
 def \_\_init\_\_(self):  
 # 定义常用变量,url,headers及计数等  
   
 def get\_html(self):  
 # 获取响应内容函数,使用随机User-Agent  
   
 def parse\_html(self):  
 # 使用正则表达式来解析页面，提取数据  
   
 def save\_html(self):  
 # 将提取的数据按要求保存，csv、MySQL数据库等  
   
 def run(self):  
 # 程序入口函数，用来控制整体逻辑  
   
if \_\_name\_\_ == '\_\_main\_\_':  
 # 程序开始运行时间戳  
 start = time.time()  
 spider = xxxSpider()  
 spider.run()  
 # 程序运行结束时间戳  
 end = time.time()  
 print('执行时间:%.2f' % (end-start))

# **spider-day02笔记**

## **猫眼电影top100抓取案例**

* **爬虫需求**
* 【1】确定URL地址  
   百度搜索 - 猫眼电影 - 榜单 - top100榜  
    
  【2】 爬取目标  
   所有电影的 电影名称、主演、上映时间
* **爬虫实现**
* 【1】查看网页源码，确认数据来源  
   响应内容中存在所需抓取数据 - 电影名称、主演、上映时间  
    
  【2】翻页寻找URL地址规律  
   第1页：https://maoyan.com/board/4?offset=0  
   第2页：https://maoyan.com/board/4?offset=10  
   第n页：offset=(n-1)\*10  
    
  【3】编写正则表达式  
   <div class="movie-item-info">.\*?title="(.\*?)".\*?class="star">(.\*?)</p>.\*?releasetime">(.\*?)</p>  
     
  【4】开干吧兄弟
* **代码实现 - 初始代码**
* """  
  抓取猫眼电影 - 榜单 - top100榜（名称、主演、上映时间）  
  """  
  import requests  
  import re  
  import time  
  import random  
    
  class MaoyanSpider:  
   def \_\_init\_\_(self):  
   self.url = 'https://maoyan.com/board/4?offset={}'  
   # 猫眼电影反爬：使用较老的User-Agent会跳到验证中心  
   self.headers = {  
   'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36'  
   }  
   # 计数  
   self.i = 0  
    
   def get\_html(self, url):  
   html = requests.get(url=url, headers=self.headers).text  
   # 直接调用解析函数  
   self.parse\_html(html)  
    
   def parse\_html(self, html):  
   """正则解析提取具体电影数据"""  
   regex = '<div class="movie-item-info">.\*?title="(.\*?)".\*?class="star">(.\*?)</p>.\*?releasetime">(.\*?)</p>'  
   pattern = re.compile(regex,re.S)  
   # r\_list: [('大圣娶亲','周星驰','1993'),('月光宝盒','周星驰','1994'),(),(),...()]  
   r\_list = pattern.findall(html)  
   # 直接调用数据处理函数  
   self.save\_html(r\_list)  
    
   def save\_html(self, r\_list):  
   """处理具体电影数据"""  
   item = {}  
   for r in r\_list:  
   item['name'] = r[0].strip()  
   item['star'] = r[1].strip()  
   item['time'] = r[2].strip()  
   print(item)  
   self.i += 1  
    
   def run(self):  
   """程序入口函数"""  
   for offset in range(0,91,10):  
   url = self.url.format(offset)  
   self.get\_html(url=url)  
   # 控制爬取速度: uniform(0,2) 生成0-2之间的浮点数  
   time.sleep(random.uniform(0,2))  
   print('数量:',self.i)  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   spider = MaoyanSpider()  
   spider.run()

## **数据持久化 - csv**

* **csv描述**
* 【1】作用  
   将爬取的数据存放到本地的csv文件中  
    
  【2】使用流程  
   2.1> 打开csv文件  
   2.2> 初始化写入对象  
   2.3> 写入数据(参数为列表)  
     
  【3】示例代码  
   import csv   
   with open('sky.csv','w') as f:  
   writer = csv.writer(f)  
   writer.writerow([])
* **示例**
* 【1】题目描述  
   创建 test.csv 文件，在文件中写入数据  
    
  【2】单行写入 - writerow([])方法  
   import csv  
   with open('test.csv','w') as f:  
   writer = csv.writer(f)  
   writer.writerow(['步惊云','36'])  
   writer.writerow(['超哥哥','25'])  
    
  【3】多行写入 - writerows([(),(),()]）方法  
   import csv  
   with open('test.csv','w') as f:  
   writer = csv.writer(f)  
   writer.writerows([('聂风','36'),('秦霜','25'),('孔慈','30')])
* **练习1 - 使用 writerow() 方法将猫眼电影数据存入本地 maoyan.csv 文件**
* 【1】在 \_\_init\_\_() 中打开csv文件，因为csv文件只需要打开和关闭1次即可  
  【2】在 save\_html() 中将所抓取的数据处理成列表，使用writerow()方法写入  
  【3】在run() 中等数据抓取完成后关闭文件
* **练习2 - 使用 writerows() 方法将猫眼电影数据存入本地 maoyan.csv 文件**
* 【1】在 \_\_init\_\_() 中打开csv文件，因为csv文件只需要打开和关闭1次即可  
  【2】在 \_\_init\_\_() 中定义存储所有电影信息的空列表，用于后序写入文件  
  【3】在 save\_html() 中将所抓取的数据处理成元组，并添加到总列表中  
  【4】在run() 中等数据抓取完成一次性使用writerows()写入后关闭文件

## **数据持久化 - MySQL**

* **pymysql回顾**
* # 1. 单条插入表记录 - excute()方法  
  # 2. 多条插入表记录 - excutemany()方法  
    
  # 示例代码如下:  
  import pymysql  
    
  db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')  
  cursor = db.cursor()  
    
  ins = 'insert into filmtab values(%s,%s,%s)'  
  # 1. 单条插入表记录之 excute() 方法  
  cursor.execute(ins,['霸王别姬','张国荣','1993'])  
  # 2. 多条插入表记录之 excutemany() 方法 - 高效且节省资源  
  cursor.executemany(ins,[('大话1','周','1993'),('大话2','周','1994')])  
    
  db.commit()  
  cursor.close()  
  db.close()
* **练习 - 将电影信息存入MySQL数据库**
* 【1】提前建库建表  
  mysql -h127.0.0.1 -uroot -p123456  
  create database maoyandb charset utf8;  
  use maoyandb;  
  create table maoyantab(  
  name varchar(100),  
  star varchar(300),  
  time varchar(100)  
  )charset=utf8;  
    
  【2】 使用excute()方法将数据存入数据库 - 在初始代码基础上做如下改动  
   2.1) 在 \_\_init\_\_() 中连接数据库并创建游标对象  
   2.2) 在 save\_html() 中将所抓取的数据处理成列表，使用execute()方法写入  
   2.3) 在run() 中等数据抓取完成后关闭游标及断开数据库连接  
    
  import pymysql  
    
  def \_\_init\_\_(self):  
   # 添加代码  
   self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')  
   self.cursor = self.db.cursor()  
     
  def save\_html(self,dd\_list):  
   # 覆盖原来代码  
   ins = 'insert into maoyantab values(%s,%s,%s)'  
   for dd in dd\_list:  
   # 将每个电影信息处理成列表  
   dd\_li = [dd[0].strip(),dd[1].strip(),dd[2].strip()]  
   self.cursor.execute(ins,dd\_li)  
   self.db.commit()  
   print(dd\_li)  
   self.i += 1  
     
  def run(self):  
   # 添加代码  
   self.cursor.close()  
   self.db.close()  
     
  【3】使用excutemany()方法将数据存入数据库 - 在初始代码基础上做如下改动  
   3.1) 在 \_\_init\_\_() 中连接数据库及创建游标对象  
   3.2) 在 \_\_init\_\_() 中定义存储所有电影信息的空列表，用于后序存入数据库  
   3.3) 在 save\_html() 中将所抓取的数据处理成元组，并添加到总列表中  
   3.4) 在run() 中等数据抓取完成一次性使用executemany()写入后断开数据库  
    
  import pymysql  
    
  def \_\_init\_\_(self):  
   # 添加代码  
   self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')  
   self.cursor = self.db.cursor()  
   # 存放所有电影信息的大列表  
   self.all\_film\_list = []  
     
  def save\_html(self,dd\_list):  
   # 覆盖原来代码  
   for dd in dd\_list:  
   dd\_tuple = (dd[0].strip(),dd[1].strip(),dd[2].strip())  
   self.all\_film\_list.append(dd\_tuple)  
   self.i += 1  
     
  def run(self):  
   # 添加代码  
   ins = 'insert into maoyantab values(%s,%s,%s)'  
   self.cursor.executemany(ins,self.all\_film\_list)  
   self.db.commit()  
   self.cursor.close()  
   self.db.close()

## **数据持久化 - MongoDB**

* **MongoDB特点**
* 【1】非关系型数据库,数据以键值对方式存储  
  【2】MongoDB基于磁盘存储  
  【3】MongoDB数据类型单一,值为JSON文档,而Redis基于内存,  
   3.1> MySQL数据类型：数值类型、字符类型、日期时间类型、枚举类型  
   3.2> Redis数据类型：字符串、列表、哈希、集合、有序集合  
   3.3> MongoDB数据类型：值为JSON文档  
  【4】MongoDB: 库 -> 集合 -> 文档  
   MySQL : 库 -> 表 -> 表记录
* **MongoDB常用命令**
* Linux进入: mongo  
  >show dbs - 查看所有库  
  >use 库名 - 切换库  
  >show collections - 查看当前库中所有集合  
  >db.集合名.find().pretty() - 查看集合中文档  
  >db.集合名.count() - 统计文档条数  
  >db.集合名.drop() - 删除集合  
  >db.dropDatabase() - 删除当前库
* **pymongo回顾**
* import pymongo  
    
  # 1.连接对象  
  conn = pymongo.MongoClient(host = 'localhost',port = 27017)  
  # 2.库对象  
  db = conn['maoyandb']  
  # 3.集合对象  
  myset = db['maoyanset']  
  # 4.插入数据库  
  myset.insert\_one({'name':'赵敏'})  
  myset.insert\_many([{'name':'小昭'},{'age':'30'}])
* **练习 - 将电影信息存入MongoDB数据库**
* """在初始代码基础上做如下更改"""  
  import pymongo  
    
  def \_\_init\_\_(self):  
   # 添加  
   self.conn = pymongo.MongoClient('localhost',27017)  
   self.db = self.conn['maoyandb']  
   self.myset = self.db['maoyanset']  
    
  def save\_html(self,r\_list):  
   # 覆盖  
   # 将数据处理为字典,执行insert\_one()  
   for r in r\_list:  
   item = {}  
   item['name'] = r[0].strip()  
   item['star'] = r[1].strip()  
   item['time'] = r[2].strip()  
   self.myset.insert\_one(item)

## **汽车之家数据抓取 - 二级页面**

* **领取任务**
* 【1】爬取地址  
   汽车之家 - 二手车 - 价格从低到高  
   https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp1exx0/  
    
     
  【2】爬取目标  
   所有汽车儿得 型号、行驶里程、上牌时间、档位、排量、车辆所在地、价格  
    
  【3】爬取分析  
   \*\*\*\*\*\*\*\*\*一级页面需抓取\*\*\*\*\*\*\*\*\*\*\*  
   1、车辆详情页的链接  
     
   \*\*\*\*\*\*\*\*\*二级页面需抓取\*\*\*\*\*\*\*\*\*\*\*  
   1、名称  
   2、行驶里程  
   3、上牌时间  
   4、档位  
   5、排量  
   6、车辆所在地  
   7、价格
* **实现步骤**
* 【1】确定响应内容中是否存在所需抓取数据 - 存在  
    
  【2】找URL地址规律  
   第1页: https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp1exx0/  
   第2页: https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp2exx0/  
   第n页: https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp{}exx0/  
     
  【3】 写正则表达式  
   一级页面正则表达式:<li class="cards-li list-photo-li".\*?<a href="(.\*?)".\*?</li>  
   二级页面正则表达式:<div class="car-box">.\*?<h3 class="car-brand-name">(.\*?)</h3>.\*?<ul class="brand-unit-item fn-clear">.\*?<li>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<span class="price" id="overlayPrice">￥(.\*?)<b>  
    
  【4】代码实现
* **代码实现**
* import requests  
  import re  
  import time  
  import random  
    
    
  class CarSpider(object):  
   def \_\_init\_\_(self):  
   self.url = 'https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp{}exx0/'  
   self.headers = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}  
    
   # 功能函数1 - 获取响应内容  
   def get\_html(self,url):  
   html = requests.get(url=url,headers=self.headers).text  
    
   return html  
    
    
   # 功能函数2 - 正则解析  
   def re\_func(self,regex,html):  
   pattern = re.compile(regex,re.S)  
   r\_list = pattern.findall(html)  
    
   return r\_list  
    
   # 爬虫函数开始  
   def parse\_html(self,one\_url):  
   one\_html = self.get\_html(one\_url)  
   one\_regex = '<li class="cards-li list-photo-li".\*?<a href="(.\*?)".\*?</li>'  
   href\_list = self.re\_func(one\_regex,one\_html)  
   for href in href\_list:  
   # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息  
   url = 'https://www.che168.com' + href  
    
   # 获取一辆汽车的信息  
   self.get\_data(url)  
   time.sleep(random.randint(1,2))  
    
   # 获取一辆汽车信息  
   def get\_data(self,url):  
   two\_html = self.get\_html(url)  
   two\_regex = '<div class="car-box">.\*?<h3 class="car-brand-name">(.\*?)</h3>.\*?<ul class="brand-unit-item fn-clear">.\*?<li>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<span class="price" id="overlayPrice">￥(.\*?)<b'  
   item = {}  
   car\_info\_list = self.re\_func(two\_regex,two\_html)  
   item['name'] = car\_info\_list[0][0]  
   item['km'] = car\_info\_list[0][1]  
   item['year'] = car\_info\_list[0][2]  
   item['type'] = car\_info\_list[0][3].split('/')[0]  
   item['displacement'] = car\_info\_list[0][3].split('/')[1]  
   item['city'] = car\_info\_list[0][4]  
   item['price'] = car\_info\_list[0][5]  
   print(item)  
    
   def run(self):  
   for p in range(1,11):  
   url = self.url.format(p)  
   self.parse\_html(url)  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   spider = CarSpider()  
   spider.run()
* **扩展 - 增量爬取**
* # 将数据存入MySQL数据库 - 增量爬取  
  【1】思路  
   1.1 MySQL中新建表 urltab,存储所有爬取过的链接的指纹  
   1.2 在爬取之前,先判断该指纹是否爬取过,如果爬取过,则不再继续爬取  
    
  【2】建库建表  
   create database cardb charset utf8;  
   use cardb;  
   create table request\_finger(  
   finger char(32)  
   )charset=utf8;  
   create table cartab(  
   name varchar(100),  
   km varchar(50),  
   years varchar(50),  
   type varchar(50),  
   displacement varchar(50),  
   city varchar(50),  
   price varchar(50)  
   )charset=utf8;
* **增量爬取 - MySQL**
* import requests  
  import re  
  import time  
  import random  
  import pymysql  
  from hashlib import md5  
  import sys  
    
    
  class CarSpider(object):  
   def \_\_init\_\_(self):  
   self.url = 'https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp{}exx0/'  
   self.headers = {  
   'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}  
   self.db = pymysql.connect('localhost', 'root', '123456', 'cardb', charset='utf8')  
   self.cursor = self.db.cursor()  
    
   # 功能函数1 - 获取响应内容  
   def get\_html(self, url):  
   html = requests.get(url=url,headers=self.headers).text  
    
   return html  
    
   # 功能函数2 - 正则解析  
   def re\_func(self, regex, html):  
   pattern = re.compile(regex, re.S)  
   r\_list = pattern.findall(html)  
    
   return r\_list  
    
   # 爬虫函数开始  
   def parse\_html(self, one\_url):  
   one\_html = self.get\_html(one\_url)  
   one\_regex = '<li class="cards-li list-photo-li".\*?<a href="(.\*?)".\*?</li>'  
   href\_list = self.re\_func(one\_regex, one\_html)  
   for href in href\_list:  
   # 加密指纹  
   s = md5()  
   s.update(href.encode())  
   finger = s.hexdigest()  
   # 如果指纹表中不存在  
   if self.go\_spider(finger):  
   # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息  
   url = 'https://www.che168.com' + href  
    
   # 获取一辆汽车的信息  
   self.get\_data(url)  
   ins = 'insert into request\_finger values(%s)'  
   self.cursor.execute(ins, [finger])  
   self.db.commit()  
   time.sleep(random.randint(1, 2))  
   else:  
   sys.exit('抓取结束')  
    
   # 判断是否存在：存在返回False，不存在返回True  
   def go\_spider(self, finger):  
   sel = 'select \* from request\_finger where finger=%s'  
   result = self.cursor.execute(sel, [finger])  
   if result:  
   return False  
   return True  
    
   # 获取一辆汽车信息  
   def get\_data(self, url):  
   two\_html = self.get\_html(url)  
   two\_regex = '<div class="car-box">.\*?<h3 class="car-brand-name">(.\*?)</h3>.\*?<ul class="brand-unit-item fn-clear">.\*?<li>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<span class="price" id="overlayPrice">￥(.\*?)<b'  
   item = {}  
   car\_info\_list = self.re\_func(two\_regex, two\_html)  
   item['name'] = car\_info\_list[0][0]  
   item['km'] = car\_info\_list[0][1]  
   item['year'] = car\_info\_list[0][2]  
   item['type'] = car\_info\_list[0][3].split('/')[0]  
   item['displacement'] = car\_info\_list[0][3].split('/')[1]  
   item['city'] = car\_info\_list[0][4]  
   item['price'] = car\_info\_list[0][5]  
   print(item)  
    
   one\_car\_list = [  
   item['name'],  
   item['km'],  
   item['year'],  
   item['type'],  
   item['displacement'],  
   item['city'],  
   item['price']  
   ]  
   ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s)'  
   self.cursor.execute(ins, one\_car\_list)  
   self.db.commit()  
    
   def run(self):  
   for p in range(1, 2):  
   url = self.url.format(p)  
   self.parse\_html(url)  
    
   # 断开数据库链接  
   self.cursor.close()  
   self.db.close()  
    
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   spider = CarSpider()  
   spider.run()
* **能不能使用redis来实现增量**
* """  
   提示: 使用redis中的集合,sadd()方法,添加成功返回1,否则返回0  
   请各位大佬忽略掉下面代码,自己独立实现  
  """  
    
  import requests  
  import re  
  import time  
  import random  
  import pymysql  
  from hashlib import md5  
  import sys  
  import redis  
    
    
  class CarSpider(object):  
   def \_\_init\_\_(self):  
   self.url = 'https://www.che168.com/beijing/a0\_0msdgscncgpi1lto1csp{}exx0/'  
   self.headers = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}  
   self.db = pymysql.connect('localhost','root','123456','cardb',charset='utf8')  
   self.cursor = self.db.cursor()  
   # 连接redis去重  
   self.r = redis.Redis(host='localhost',port=6379,db=0)  
    
   # 功能函数1 - 获取响应内容  
   def get\_html(self,url):  
   html = requests.get(url=url,headers=self.headers).text  
    
   return html  
    
   # 功能函数2 - 正则解析  
   def re\_func(self,regex,html):  
   pattern = re.compile(regex,re.S)  
   r\_list = pattern.findall(html)  
    
   return r\_list  
    
   # 爬虫函数开始  
   def parse\_html(self,one\_url):  
   one\_html = self.get\_html(one\_url)  
   one\_regex = '<li class="cards-li list-photo-li".\*?<a href="(.\*?)".\*?</li>'  
   href\_list = self.re\_func(one\_regex,one\_html)  
   for href in href\_list:  
   # 加密指纹  
   s = md5()  
   s.update(href.encode())  
   finger = s.hexdigest()  
   # 如果指纹表中不存在  
   if self.r.sadd('car:urls',finger):  
   # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息  
   url = 'https://www.che168.com' + href  
    
   # 获取一辆汽车的信息  
   self.get\_data(url)  
   time.sleep(random.randint(1,2))  
   else:  
   sys.exit('抓取结束')  
    
   # 判断是否存在：存在返回False，不存在返回True  
   def go\_spider(self,finger):  
   sel = 'select \* from request\_finger where finger=%s'  
   result = self.cursor.execute(sel,[finger])  
   if result:  
   return False  
   return True  
    
   # 获取一辆汽车信息  
   def get\_data(self,url):  
   two\_html = self.get\_html(url)  
   two\_regex = '<div class="car-box">.\*?<h3 class="car-brand-name">(.\*?)</h3>.\*?<ul class="brand-unit-item fn-clear">.\*?<li>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<h4>(.\*?)</h4>.\*?<span class="price" id="overlayPrice">￥(.\*?)<b'  
   item = {}  
   car\_info\_list = self.re\_func(two\_regex,two\_html)  
   item['name'] = car\_info\_list[0][0]  
   item['km'] = car\_info\_list[0][1]  
   item['year'] = car\_info\_list[0][2]  
   item['type'] = car\_info\_list[0][3].split('/')[0]  
   item['displacement'] = car\_info\_list[0][3].split('/')[1]  
   item['city'] = car\_info\_list[0][4]  
   item['price'] = car\_info\_list[0][5]  
   print(item)  
    
   one\_car\_list = [  
   item['name'],  
   item['km'],  
   item['year'],  
   item['type'],  
   item['displacement'],  
   item['city'],  
   item['price']  
   ]  
   ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s)'  
   self.cursor.execute(ins,one\_car\_list)  
   self.db.commit()  
    
   def run(self):  
   for p in range(1,2):  
   url = self.url.format(p)  
   self.parse\_html(url)  
    
   # 断开数据库链接  
   self.cursor.close()  
   self.db.close()  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   spider = CarSpider()  
   spider.run()