1. super（）的使用以及为什么要使用

**super使用的前提：**

1. 两个类存在继承关系情况

**super使用场景**

1. 子类使用父类定义的属性或方法时

**Super所使用的位置**

1. 只能出现在子类中

**注意事项：**

4）\_\_init\_\_()函数在Python的类中有且只有一个，不能重载（在Python中没有重载的概念，只覆盖）

C语言--面向过程的编程语言--》提取函数

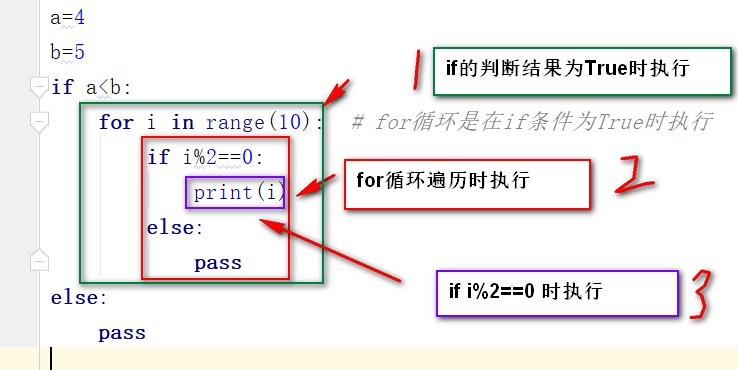
Java语言--面向对象的编程语言--》重点强调 类的封装 (根据N多个对象提取出类)

Python语言--面向对象的编程语言--平分秋色（可以提取函数以面向过程的方式编写代码 ， 还可以进类的封装，以面向对象的方式编写代码）

|  |
| --- |
| *# 教育机构：马士兵教育 # 讲 师：杨淑娟* **class** Person():  **def** \_\_init\_\_(self,name,age): *#name 是局变量，age也是局部变量* self.xm=name *# 左侧的self.xm 才是类的属性* self.nl=age *#左侧的self.nl才是类的属性* **class** Student(Person):   **def** \_\_init\_\_(self,name,age,stuno): *#self相当于Java语言中的this* super().\_\_init\_\_(name,age) *# self只作为函数定义时参数，函数调用时无需传参* self.xh=stuno   stu=Student(**'marry'**,20,**'msb1001'**) print(stu.xm) *# 对象名.属性名* print(stu.nl) |

1. If嵌套语句内含多层结构，初学者把握其中关联与节点，比较迷茫？

在Python中有“严格”的缩进，来控制程序的层次结构



1. 其次，反爬应对中的JS逆向与token思维,能否引导漫谈？
2. 教下fidder抓包教程
3. 从固定银行卡批量向其他固定银行卡转账不同金额，过程分析不出来，分析一下上述问题实现的过程

|  |
| --- |
| *# 教育机构：马士兵教育 # 讲 师：杨淑娟  #函数的调用，通过函数名调用* **def** show():  print(**'''helloworld'''**)  show() *# 调用函数* **import** openpyxl filename=**'salary.xlsx'** *#模拟银行发工资 def是定义函数的关键字* **def** getsalary(cardno,account,salary):  *# 在Python中格式化字符串有三种方式，% ，format函数，简写的f* print(**f'账户名称为{**account**}的{**cardno**}实发工资{**salary**}元'**)  **def** getemp(filename):  lst=[]  wk=openpyxl.load\_workbook(filename)  sheet=wk.active *# 获取当前活动的sheet页* rows=sheet[**'A2:C4'**]  **for** row **in** rows:  sub\_lst=[]  **for** cell **in** row:  sub\_lst.append(cell.value)  lst.append(sub\_lst)  **return** lst  **def** start():  lst=getemp(filename) *#调用获取员工列表的函数* **for** item **in** lst: *#每一个item就是列表中一个员工的数据* getsalary(item[0],item[1],item[2]) *#调用获取薪水的函数   #以主函数的形式运行* **if** \_\_name\_\_ == **'\_\_main\_\_'**:  start() |

1. 开发QQ群发助手 迷宫游戏的路径选优（队列使用） （肖老师） 车牌人脸识别思路及算法（卢老师）
2. a=30，a+=20得出a=50，不太能理解
3. 想知道在pycharm里打代码时出现红色波浪线是什么意思
4. requests库和urllib.request的区别，以及各自的应用场景有什么不同吗？

Requests库是Python的第三方库 ，更好用一些

urllib.request是Python自带的

(安装完操作系统会自带一个IE浏览器，但是我们通常都会下载前安装其它的浏览，比如说谷歌，360等)

1. 数组的应用解例 （Python中没有数组的概念，与数组相同的数据结构叫列表）
2. 爬取资源卡死或蓝屏（具体代码） 讲解下HTML是啥**（前端）**

**（https://ke.qq.com/course/3062654）**

|  |
| --- |
|  |

**学会看异常类型 （XXXError就是异常的类型，还要学会去看产生异常的位置）**

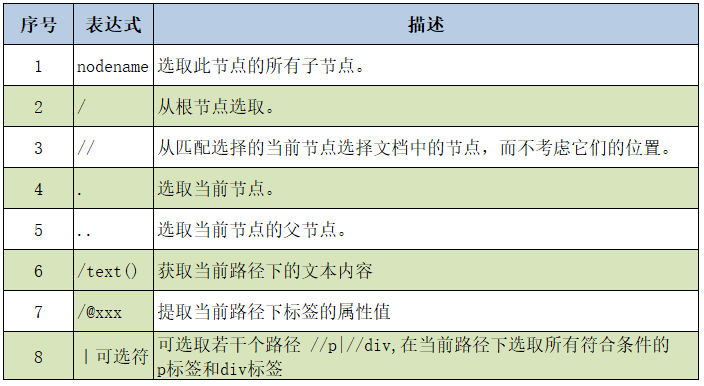
1. 正则中的match函数是指判断一个字符串能否从起始处全部或者部分地匹配，其中是如何进行匹配，是正向索引依次进行判断还是有内在逻辑（肖老师直播答疑的视频）
2. 为什么使用Decimal提升浮点数计算精度后用变量相加还是会很长的小数长度，如果不行，那如何令变量中的浮点数计算后仍然的精确的（round()函数）
3. 1）电子邮件的附件怎么批量下载 2）有的网站数据 不对外 可以爬取吗？（不对外，不能） 3）PDF文档的密码如果不知道，可以解密吗？（不能）
4. 如何保存文件（保存什么文件） 爬虫兼职指导（关注老师朋友圈）

|  |
| --- |
| 2 |

1. 物联网怎么样 想学习做这方面的事（全栈中有物联网的项目）
2. 课程学到什么程度才能就业，或者跟着老师一起做项目？（学到全栈的项目）
3. 详细讲讲多线程的实际应用（肖老师直播答疑）
4. 人工智能，学完python基础后，该下一步呢学AI还是什么（卢老师）
5. ocl模块安装出错（具体的异常类型）
6. xpath一些方面相关的东西，总有点不明白（不明白具体）

Xpath是解析html的一个工具

怎么去安装xpath? Python 100问第60





1. 爬取网站信息时遇到反爬怎么处理

反反爬的主要思路就是模拟浏览器，浏览器如何操作，代码中就如何去实现

例如：浏览器先请求了地址url1，保留了cookie在本地，之后请求地址url2，带上了之前的cookie，代码中也可以这样去实现。

反反爬的主要思路

反反爬的主要思路就是：尽可能的去模拟浏览器，浏览器在如何操作，代码中就如何去实现。浏览器先请求了地址url1，保留了cookie在本地，之后请求地址url2，带上了之前的cookie，代码中也可以这样去实现。

很多时候，爬虫中携带的headers字段，cookie字段，url参数，post的参数很多，不清楚哪些有用，哪些没用的情况下，只能够去尝试，因为每个网站都是不相同的。当然在盲目尝试之前，可以参考别人的思路，我们自己也应该有一套尝试的流程。

1. 通过headers字段来反爬

1.1 通过headers中的User-Agent字段来反爬

通过User-Agent字段反爬的话，只需要给他在请求之前添加User-Agent即可，更好的方式是使用User-Agent池来解决,我们可以考虑收集一堆User-Agent的方式，或者是随机生成User-Agent

1.2 通过referer字段或者是其他字段来反爬

通过referer字段来反爬，我们只需要添加上即可

1.3 通过cookie来反爬

如果目标网站不需要登录 每次请求带上前一次返回的cookie，比如requests模块的session

如果目标网站需要登录 准备多个账号，通过一个程序获取账号对应的cookie，组成cookie池，其他程序使用这些cookie

2.通过js来反爬

2.1 通过js实现跳转来反爬

在请求目标网站的时候，我们看到的似乎就请求了一个网站，然而实际上在成功请求目标网站之前，中间可能有通过js实现的跳转，我们肉眼不可见，这个时候可以通过点击perserve log按钮实现观察页面跳转情况

在这些请求中，如果请求数量很多，一般来讲，只有那些response中带cookie字段的请求是有用的，意味着通过这个请求，对方服务器有设置cookie到本地

2.2 通过js生成了请求参数

对应的需要分析js，观察加密的实现过程

可以使用selenium模块解决

3.3 通过js实现了数据的加密

对应的需要分析js，观察加密的实现过程,也是使用selenium模块实现

3. 通过验证码来反爬

通过打码平台或者是机器学习的方法识别验证码，其中打码平台廉价易用，建议使用

4. 通过ip地址来反爬

同一个ip大量请求了对方服务器，有更大的可能性会被识别为爬虫，对应的通过购买高质量的ip的方式能够解决

5. 其他的反爬方式

6.1 通过自定义字体来反爬

解决思路：可以尝试切换到手机版试试

6.2 通过css来反爬

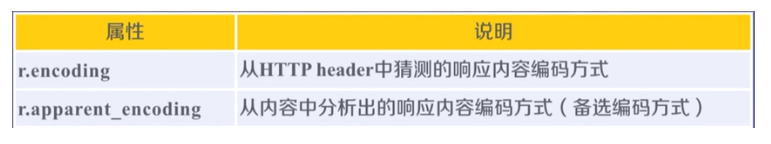
如通过css掩盖真实数据

解决思路：计算css的偏量



|  |
| --- |
| **import** requests **import** sys **import** io  url=**'https://kyfw.12306.cn/otn/leftTicket/queryT?leftTicketDTO.train\_date=2021-03-19&leftTicketDTO.from\_station=CCT&leftTicketDTO.to\_station=BJP&purpose\_codes=ADULT'** headers={**'User-Agent'**: **'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36'**,  **'cookie'**:**'JSESSIONID=E3C3A2944BD8D6D0A342F585649A5DD4;\_jc\_save\_fromDate=2021-03-19'**,   **'Referer'**: **'https: // kyfw.12306.cn / otn / leftTicket / init?linktypeid = dc & fs = % E5 % 8C % 97 % E4 % BA % AC, BJP & ts = % E4 % B8 % 8A % E6 % B5 % B7, SHH & date = 2021 - 03 - 12 & flag = N, N, Y'** }  resp=requests.get(url,headers=headers) json\_ticket = resp.json() data\_lst = json\_ticket[**'data'**][**'result'**] **for** item **in** data\_lst:  d = item.split(**'|'**)  print(d[3], d[6], d[7], d[31], d[30], d[13]) *# 3为车次 6为始发站，7为终点站* |

# **Python爬虫乱码问题之encoding和apparent\_encoding的区别**



encoding是从http中的header中的charset字段中提取的编码方式，若header中没有charset字段则默认为ISO-8859-1编码模式，则无法解析中文，这是乱码的原因

apparent\_encoding会从网页的内容中分析网页编码的方式，所以apparent\_encoding比encoding更加准确。当网页出现乱码时可以把apparent\_encoding的编码格式赋值给encoding。

1. 层次化索引方式
2. 老师具体说一下，关于爬虫哪些内容可以爬取，哪些明确不行，哪些内容模糊没有明确
3. 讲解一下用filter函数的高阶用法来求100以内的质数的方法

|  |
| --- |
| filter(function or None, iterable) 筛选可迭代对象iterable 中的数据，返回一个可迭代对象，此可迭代对象将对iterable生成的数据进行筛选    function 数将对iterable中每个元素进行求值，返回Flase则将此数据丢充，      返回True则保留此数据    示例：      # 写一个函数判断是奇数还是偶数：      def isodd(x):          return x % 2 == 1      # odd = [x for x in range(10) if isodd(x)]      odd = [x for x in filter(isodd, range(10))]    # 2. 用filter函数将1~100 所有素数求和后打印出来      def isprimes(x):          for i in range(2, x):              if x % i == 0:                  return False          return True      # print([x for x in filter(isprimes, range(1, 101))])      print(sum(filter(isprimes, range(1, 101)))) |

1. 爬取B站视频
2. b站番剧的评论爬取失败，老师演示一下
3. 讲一个爬小说和爬电影的代码案例？
4. 办公自动化的三方模块只有课程里的吗？还有没别的模块
5. ①什么时候需要写上encoding=' '，在使用encoding=' '的时候，什么时候填GBK什么时候填UTF-8 ②学习Python需要学习计算机相关知识或底层原理吗
6. 递归函数的调用过程