# 遇到问题怎么办：

IDLE – Help 索引中

PEP

<https://www.python.org/dev/peps/>

Import 模块 print(模块.\_\_doc\_\_)

Dir(timeit)看看“模块.\_\_all\_\_”哪些是可以外部调用的

From 模块 import \* 会导入\_\_all\_\_中的模块

模块.\_\_file\_\_可以看到源代码

Help(模块)

# 爬虫

基本爬虫（用有道翻译翻译内容）

到network那里，点击

寻找Post的方法

Post是发送提交数据

Get是请求获得数据

Post中preview有翻译结果

Headers中的Request URL才是真正打开的url

Request Header- User Agent（可自定义）是判断是否非人类访问

Form Data就是要提交的内容

把form data 转化成字典

Urllib.request.urlopen(url, data=None,)

如果data没有东西，Python将用get方法与服务端发生关系

否则用Post 方法。

Data 必须以某种格式提交，用 urllib.parse.urlencode()可以转换

注意必须.encode(“utf-8”)

# 爬虫

1. 踩点 发现url的规律
2. 进入主页时审查元素可发现span中的current-comment-page
3. 审查元素发现img元素 图片地址。

# JSON模块

在数据传递方面，XML是一种选择，还有一种选择是JSON，它是一种轻量级的数据交换格式。

[Python](http://lib.csdn.net/base/python)标准库中有JSON模块，主要执行序列化和反序列化的功能。

序列化 dumps()：encoding，把一个python队形编码转化成JSON字符串，JSON字符串是字符串

反序列化 loads()：decoding，把JSON格式字符串解码转换为python数据对象，即为字典对象

dumps()

dumps函数的功能：转换为字符串的格式

语法格式：   
dumps(data, sort\_keys = True, indent = 2)   
其中，sort\_keys = True 意思是按照键的字典顺序排序，indent = 2是让每个键/值对显示的时候，以缩进两个字符对齐

例子：

>>> import json

>>> data = [{"name":"xiaoxu","lang":("python","english"),"age":40}]

>>> data\_json = json.dumps(data)

>>> print data\_json

[{"lang": ["python", "english"], "age": 40, "name": "xiaoxu"}]

>>>

>>>

>>> data\_j = json.dumps(data,sort\_keys = True,indent =2)

>>> print data\_j

[

{

"age": 40,

"lang": [

"python",

"english"

],

"name": "xiaoxu"

}

]

>>>

注意：观察data和data\_json有两点不同

lang的值从元组变成了列表。

看如下结果   
>>> print data\_json   
[{"lang": ["python", "english"], "age": 40, "name": "xiaoxu"}]   
>>> type(data\_json)   
<type 'str'>   
>>> type(data)   
<type 'list'>   
>>>

loads()

loads 函数的功能：转换为字典的格式

>>> import json

>>>

>>> print data\_json

[{"lang": ["python", "english"], "age": 40, "name": "xiaoxu"}] #接上个代码的结果输出

>>>

>>> new\_data = json.loads(data\_json)

>>> print new\_data

[{u'lang': [u'python', u'english'], u'age': 40, u'name': u'xiaoxu'}]

>>>

注意：解码之后并没有将元组还原

若数据不是很大，那么上面的操作足够了。前面的操作方法是将数据都读入内存，若数据量太大了内存会爆满，这肯定是不行的。针对此问题，JSON提供了load()函数和dump()函数来解决这个问题。

例：

>>> import tempfile

>>> data

[{'lang': ('python', 'english'), 'age': 40, 'name': 'xiaoxu'}]

>>> f = tempfile.NamedTemporaryFile(mode = 'w+')

>>> json.dump(data,f)

>>> f.flush()

>>> print open(f.name,"r").read()

[{"lang": ["python", "english"], "age": 40, "name": "xiaoxu"}]

2.如何隐藏Python的爬虫身份。

Headers的User Agent

通过Request的headers参数修改

通过Request.add\_header()方法

延迟提交时间

time.sleep(6)

使用代理

1. 参数是一个字典｛’类型’：‘代理ip:端口号’｝

proxy\_support = urllib.request.ProxyHandler({})

1. 定制、创建opener

opener = urllib.request.build\_opener(proxy\_support)

1. 安装opener

urllib.request.install\_opener(opener)

或opener.open(url)

# BeautifulSoup模块

下面的一段HTML代码将作为例子被多次用到.这是 爱丽丝梦游仙境的 的一段内容(以后内容中简称为 爱丽丝 的文档):

html\_doc = """

<html><head><title>The Dormouse's story</title></head>

<body>

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were

<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,

<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

<p class="story">...</p>

"""

使用BeautifulSoup解析这段代码,能够得到一个 BeautifulSoup 的对象,并能按照标准的缩进格式的结构输出:

from bs4 import BeautifulSoup

soup = BeautifulSoup(html\_doc)

print(soup.prettify())

# <html>

# <head>

# <title>

# The Dormouse's story

# </title>

# </head>

# <body>

# <p class="title">

# <b>

# The Dormouse's story

# </b>

# </p>

# <p class="story">

# Once upon a time there were three little sisters; and their names were

# <a class="sister" href="http://example.com/elsie" id="link1">

# Elsie

# </a>

# ,

# <a class="sister" href="http://example.com/lacie" id="link2">

# Lacie

# </a>

# and

# <a class="sister" href="http://example.com/tillie" id="link2">

# Tillie

# </a>

# ; and they lived at the bottom of a well.

# </p>

# <p class="story">

# ...

# </p>

# </body>

# </html>

几个简单的浏览结构化数据的方法:

soup.title

# <title>The Dormouse's story</title>

soup.title.name

# u'title'

soup.title.string

# u'The Dormouse's story'

soup.title.parent.name

# u'head'

soup.p

# <p class="title"><b>The Dormouse's story</b></p>

soup.p['class']

# u'title'

soup.a

# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

soup.find\_all('a')

# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,

# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,

# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.find(id="link3")

# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>

从文档中找到所有<a>标签的链接:

for link in soup.find\_all('a'):

print(link.get('href'))

# http://example.com/elsie

# http://example.com/lacie

# http://example.com/tillie

从文档中获取所有文字内容:

print(soup.get\_text())

# The Dormouse's story

#

# The Dormouse's story

#

# Once upon a time there were three little sisters; and their names were

# Elsie,

# Lacie and

# Tillie;

# and they lived at the bottom of a well.

#

# ...