# requests库

import requests

## GET

params = {

“one”:a

}

requests.get(“URL”, params=params)

## POST

data = {

“one”:a

}

# data组成格式是one=a&two=b

requests.post(“URL”, data=data)

heade = {

“user-agent”:”xxx”

}

res = requests.post(“URL”, headers=heade, data=data)

res.content

res.text

response.text()

response.json()

response.encoding() #使用服务器返回的编码

response.status\_code() # 返回状态码

content-type:application/x-www-form-urlencoded默认

对应的user=body&password=123

requests.post(“url”, json=data)

content-type:application/json

“{\”user\”:\”body\”}

requests.post(“url”, data=json.dump(data))

content-type:--

{user: body}

发送带session的请求

# requests会每次都携带cookie

session = requests.session()

session.post()

session.get()

#获取cookie，通过requests.get()发送

session = requests.session()

res = session.post()

# 2种

cookie = res.cookies

cookie = res.cookies.get\_dict() #获取字典类型的cookie

html = requests.get(URL, cookies=cookie)

# 底层库urllib

## GET

import urllib.request

re = urllib.request.Request(Url)

reponse = urllib.request.urlopen(re)

info = reponse.read().decode(“utf-8”)

## POST

# 底层方法

**Python3将库urllib2改名为urllib,并包请求方法包含在request对象**

urllib.request HTTP请求模块

-->Request 用于构造复杂的请求，通过urlopen发送

-->urlopen 可发送Request构建的请求，可直接发送请求，最简单的方法，这一个系统定义的opener模块

-->Handler http处理器，更灵活的请求方法  
urllib.error 异常处理模块

-->error.URLError 错误类型，使用try语法  
urllib.parse URL解析模块

-->urlencode 编码解析，常用  
urllib.robotparser robots.txt解析模块

## 1/ urllib.request.Request(Url,data,headers,method)

***构造一个url请求***

--URl ：请求地址

--data： base64数据，主要适用于POST请求，**需要parse编码**

--header：报文头，**字典类型**，可使用其他方法添加

--method：get、POST

--**Re：返回值**

**--导入模块：urllib.requset**

--------------------

支持方法：

Re.add\_header(“User-Agent”, header\_string) #添加header头

Re.get\_header(“User-agent”) #获取header指定信息

## 2/ urllib.request.urlopen(re,data,timeout)

***打开一个url/request对象,发送请求（可携带参数）***

--re：urllib.request.Request的返回对象/URl地址

--data：data=data请求数据，可不带（GET），携带（POST）

--timeout：设置超时时间

**--reponse：返回值**

**--导入模块：urllib.request**

**-----------------**

支持方法:

#对HTTPResponse类型数据进行操作

**reponse**.read() / .readline()/ .readlines()/ .fileno()/ .close()

#返回HTTPMessage对象，表示远程服务器返回的头信息

**reponse**.info()

**reponse**.getcode()#返回Http状态码

**reponse**.geturl() #返回请求的url

**reponse**.status # 响应状态，属性

**reponse**.msg # 信息，属性

**reponse**.version #版本，属性

#转码，读取响应数据

**reponse**.read().decode(“utf-8”)

# 附录：转码

b：bytes编码 u：unicode编码 r：非转义字符

str.encode('utf-8')  
bytes.decode('utf-8')

## 3/ parse.urlencode(form\_data,encoding=’utf8’)

***将字符串转换为url编码***

--form\_data：数据，字典

--encoding=’utf8’：编码

**--导入模块：urllib**

--------------------

注意：

Post的数据必须是bytes或者iterable of bytes；

当urllib.request.Request()函数使用时还应使用

data = bytes(urllib.parse.urlencode({'word': 'hello'}), encoding='utf8')

response = urllib.request.urlopen('http://httpbin.org/post', data=data)

**bytes(from, encode=”utf8”)**

防止出现 'str' object has no attribute 'items'

## 4/ HTTPHandler

### Handler处理器分类

1. HTTPHandler() ：没有任何特殊功能
2. ProxyHandler(普通代理)
3. ProxyBasicAuthHandler(密码管理器对象) ：私密代理

4、HTTPBasicAuthHandler(密码管理器对象) : web客户端认证

### 一般使用流程

1. 创建相关的Handler处理器对象

http\_handler = urllib.request.HTTPHandler()

2、创建自定义opener对象

opener = urllib.request.build\_opener(http\_handler)

1. 利用opener对象打开url

**req** = urllib.request.Request(url,headers=headers)

res = opener.open(**req**)

result = res.read().decode("utf-8")

### 一般使的函数

#### （1）urllib.HTTPHandler(debuglevel)

***构建一个HTTPHandler处理对象，支持HTTP请求***

--debuglevel：debug等级1，自动开启模式

**--导入模块：urllib**

**--返回值：http\_handler**

#### （2）urllib.build\_opener(http\_handler, proxy)

***构建一个处理对象，可添加多个处理器对象***

--http\_handler：HTTPHandler返回的对象

--proxy：代理处理器，proxyBasicAuthHandler对象

**--导入模块：urllib**

**--返回值：opener**

**------------**

opener**.**addheaders：添加head头，数组

#### （3）opener.open(re)

*打开一个url连接*

--re：urllib.request.Request()返回的构造对象

**--导入模块：urllib**

**--返回值：response**

**---------------**

response.read 读取数据

#### （5）ProxyHandle(disc)

***普通代理代理服务器***

--disc：字典{“http”:”127.0.0.1:80”}

使用有认证的代理{“http”:”root:密码@127.0.0.1:80”}

**--导入模块：urllib**

**--返回值：HTTP\_proxy**

#### （6）urllib.install\_opener(opener)

***创建一个全局opener对象，多个opener对象发送信息 使用***

--opener：urllib.build\_opener创建的对象

**--导入模块：urllib**

**--返回值：无**

### 密码/代理程序实现流程

***密码管理器对象作用***

**（1）私密代理（2）Web客户端认证**

***操作流程***

（1）创建密码管理器对象

pwdmg = urllib.request.HTTPPasswordMgrWithDefaultRealm()

（2）把认证信息添加到密码管理器对象

pwdmg.add\_password(None,webserver,user,passwd)

1. 创建Handler处理器对象

--->私密代理

proxy = urllib.request.ProxyAuthBasicHandler(pwdmg)

--->Web客户端

webbasic = urllib.request.HTTPBasicAuthHandler(pwdmg)

1. 使用openner
2. request构造

### 密码/代理使用的函数

#### （1）HTTPPasswordMgrWithDefaultRealm()

***密码管理对象***

**--导入模块：urllib**

**--返回值：passwordMgr**

-------------

方法：

“””

授权账户信息

@realm：域，没有天线none

@webserver：服务器地址

@root：账户

@password：密码

“””

passwordMgr.add\_password(realm, webserver, root, password)

#### （2）Urllib.HTTPBasicAuthHandler(passwordMgr)

***密码认证方法***

--passwordMgr：HTTPPasswordMgrWithDefaultRealm对象

**--导入模块：urllib**

**--返回值：http\_handler，可用于urllib.build\_opener()**

#### （2）urllib.ProxyBasicAuthHandler(passwordMgr)

***代理处理管理器类***

--passwordMgr：HTTPPasswordMgrWithDefaultRealm对象

**--导入模块：urllib**

**--返回值：proxyauth\_handler，可用于urllib.build\_opener()**

### cookie

#### （1）cookielib.cookiejar()

***构建一个cookiejar对象，用于保存cookie的值***

**--导入模块：cookie**

**--返回值：cookie对象**

#### （2）urllib.HTTPCookieProcessor(cookie)

***构建一个处理对象用于处理cookie***

--cookie：cookie.cookiejar返回的对象

**--导入模块：urllib**

**--返回值：cookie\_handler**

## 5/ error.URLError

from urllib import request, error

try:

response = request.urlopen('http://cuiqcoex.htm')

except **error.HTTPError** as e:

print(**e.reason, e.code, e.headers**, sep='\n')

except **error.URLError** as e:

print(**e.reason**)

else:

print('Request Successfully')