```
host: 合法的Internet主机域名或IP地址
                              HTTP协议采用URL作为定位网络资源的标识, URL格式如下:
                                                                          port: 端口号, 缺省端口为80
                                                                          path: 请求资源的路径
                              URL的理解: URL是通过HTTP协议存取资源的Internet路径, 一个URL对应一个数据资源
                 HTTP协议
                                                     get: 请求获取URL位置的资源
                                                    post:请求向URL位置的资源后附加新的数据
                                                     head:请求获取URL位置资源的响应消息报告,即获得该资源的头部信息
                                                     put:请求向URL位置存储一个资源,覆盖原URL位置的资源
                              HTTP协议对资源的操作方法
                                                     patch:请求局部更新URL位置的资源、即改变该外资源的部分内
                                                     delete:请求删除URL位置存储的资源
                                                    注: patch和put的区别
                                                             构造一个请求,支撑以下各方法的基础方法
                                                         获取HTML网页的主要方法,对应于HTTP的GET
                                            requests.get()
                                            requests.head() 获取HTML网页头信息的方法,对应于HTTP的HEAD
                                            requests.post() 向HTML网页提交POST请求的方法,对应于HTTP的POST
                     requests库的主要方法
                                            requests.put() 向HTML网页提交PUT请求的方法,对应于HTTP的PUT
                                            requests.patch() 向HTML网页提交局部修改请求,对应于HTTP的PATCH
                                                            向HTML页面提交删除请求,对应于HTTP的DELETE
                                             requests.delete()
                                             requests.options(): 询问可以提交哪些方法,对应于HTTP的OPTIONS
                                           requests.request(method, url, **kwarqs)
                                                                              method:请求方式
                                                                              r = requests.request('get', url, **kwargs)
                                                                              r = requests.request('head', url, **kwargs)
                                                                              r = requests.request('post', url, **kwargs)
                                           method:请求方式,对应get/put/post等7种
                                                                             r = requests.request('put', url, **kwargs)
                                                                              r = requests.request('patch', url, **kwargs)
                                                                              r = requests.request('delete', url, **kwargs)
                                                                              r = requests.request('options', url, **kwargs)
                       requests.request()
                                           url: 拟获取页面的url链接
                                                                        params:字典或字节序列,作为参数增加到url中(百度搜索的wd)
                                                                       data : 字典、字节序列或文件对象,作为Request的内容 (一般用于post请求)
                                                                       headers:字典,HTTP定制头
                                                                        timeout:设定超时时间,秒为单位
                                                                       proxies:字典类型,设定访问代理服务器,可以增加登录认证
                                                                        json: JSON格式的数据, 作为Request的内容
                                                                        cookies:字典或CookieJar, Request中的cookie
requests库
                                           **kwargs: 控制访问的参数,共13个
                                                                                                 auth = (user,pwd)
response = requests.get(url,auth=auth)
                                                                        auth : 元组,支持HTTP认证功能
                                                                        files :字典类型,传输文件
                                                                        allow_redirects : True/False,默认为True,重定向开关
                                                                        stream : True/False,默认为True,获取内容立即下载开关
                                                                        verify : True/False,默认为True,认证SSL证书开关 之前12306就有这个问题,改为False就好了
                                                                        cert : 本地SSL证书路径
                                                                           url:拟获取页面的url链接
                                       requests.get(url, params=None, **kwargs)
                                                                          params: url中的额外参数,字典或字节流格式,可选
                                                                           **kwargs: 12个控制访问的参数
                                       本质上是调用request方法: requests.request()('get', url, params=params, **kwargs)
                                                                               r.status_code: HTTP请求的返回状态,200表示连接成功,404表示失败
                                                                               r.text: HTTP响应内容的字符串形式,即,url对应的页面内容
                                                                              r.encoding: 从HTTP header中猜测的响应内容编码方式
                      requests.get()
                                      r = requests.get(url) , 这里返回的r是Response对象
                                                                              r.apparent_encoding : 从内容中分析出的响应内容编码方式 (备选编码方式)
                                                                              r.content : HTTP响应内容的二进制形式(可以解码, 即.decode('utf-8))
                                                                              注: encoding和apparent_encoding的区别
                                                                              如果返回的是一个json文件,则r.json()可以获得这个json文件
                                                                                             proxy = { 'http':'116.209.54.154:9999'}
r = requests.get(url,proxies = proxy,timeout = 10)
                                                在请求的方法中 (比如get或者post) 传递proxies参数就可以
                                       requests.post(url, data=None, json=None, **kwargs)
                                      url: 拟更新页面的url链接
                                      data:字典、字节序列或文件, Request的内容
                    requests.post()
                                      json: JSON格式的数据,Request的内容
                                      **kwargs: 11个控制访问的参数
                                                                                   session = requests.session()
                                        requests库的session对象能够帮我们跨请求保持某些参数,
                                                                                   session.post(登陆界面的那个url.data=data.headers=headers)
                                        也会在同一个session实例发出的所有请求之间保持cookies。
                                                                                   resp = session.get(登录后需要爬取的url)
                   requests.session()
                                       def getHTMLtext(url,headers,timeout=5):
                                        except:
return
                  一个通用的代码框架
```

HTTP, Hypertext Transfer Protocol, 超文本传输协议

HTTP是一个基于"请求与响应"模式的、无状态的应用层协议

注: patch和put的区别

假设URL位置有一组数据UserInfo,包括UserID、UserName等20个字段需求:用户修改了UserName,其他不变

- 采用PATCH, 仅向URL提交UserName的局部更新请求
- 采用PUT,必须将所有20个字段一并提交到URL,未提交字段被删除 PATCH的最主要好处:节省网络带宽

注: encoding和apparent encoding的区别

r.encoding: 如果header中不存在charset,则认为编码为ISO-8859-1 r.text根据r.encoding显示网页内容 (有可能会出错) r.apparent_encoding: 根据网页内容分析出的编码方式,准确性较高,但是

requests.session()的用法

分析需要时间

```
import requests
url = "http://www.renren.com/PLogin.do"
data =
{"email":"97138074@qq.com",'password':"pythonspider"}
headers = {
    'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/62.0.3202.94 Safari/537.36"
}
# 登录
session = requests.session()
session.post(url,data=data,headers=headers)
# 访问大鹏个人中心
resp = session.get('http://www.renren.com/880151247/
profile')
print(resp.text)
```