网络爬虫（又被称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。

根据使用场景，网络爬虫可分为 通用爬虫 和 聚焦爬虫 两种.

通用爬虫

通用网络爬虫是捜索引擎抓取系统（Baidu、Google、Yahoo等）的重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。

通用搜索引擎（Search Engine）工作原理

通用网络爬虫，从互联网中搜集网页，采集信息，这些网页信息用于为搜索引擎建立索引从而提供支持，它决定着整个引擎系统的内容是否丰富，信息是否即时，因此其性能的优劣直接影响着搜索引擎的效果。

通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询，无法准确理解用户的具体需求。

Robots协议（也叫爬虫协议、机器人协议等），全称是“网络爬虫排除标准”（Robots Exclusion Protocol），网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取



聚焦爬虫

聚焦爬虫，是"面向特定主题需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： 聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。

HTTP和HTTPS

HTTP协议（HyperText Transfer Protocol，超文本传输协议）：是一种发布和接收 HTML页面的方法。

HTTPS（Hypertext Transfer Protocol over Secure Socket Layer）简单讲是HTTP的安全版，在HTTP下加入SSL层。

SSL（Secure Sockets Layer 安全套接层）主要用于Web的安全传输协议，在传输层对网络连接进行加密，保障在Internet上数据传输的安全。

HTTP的端口号为80，HTTPS的端口号为443

URL（Uniform / Universal Resource Locator的缩写）：统一资源定位符，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。

基本格式：scheme://host[:port#]/path/…/[?query-string][#anchor]

scheme：协议(例如：http, https, ftp)

host：服务器的IP地址或者域名

port#：服务器的端口（如果是走协议默认端口，缺省端口80）

path：访问资源的路径

query-string：参数，发送给http服务器的数据

anchor：锚（跳转到网页的指定锚点位置）

urllib库

urlopen( )

网页抓取数据

from urllib import request  
if \_\_name\_\_ == "\_\_main\_\_":  
 # 向指定的url发送请求，并返回服务器响应的类文件对象  
 response = request.urlopen("https://my.alipay.com/portal/i.htm")  
 html = response.read()  
 print(html.decode('utf-8'))

urlopen()返回的对象，可以使用read()进行读取，同样也可以使用geturl()方法、info()方法、getcode()方法。

geturl()返回的是一个url的字符串；

info()返回的是一些meta标记的元信息，包括一些服务器的信息；

getcode()返回的是HTTP的状态码，如果返回200表示请求成功。

向服务器发送数据

使用data参数，向服务器发送数据。GET用于信息获取，POST向服务器提交数据

如果没有设置urlopen()函数的data参数，HTTP请求采用GET方式，也就是从服务器获取信息，如果设置data参数，HTTP请求采用POST方式，也就是我们向服务器传递数据。

data参数有自己的格式，使用urllib.parse.urlencode()函数将字符串自动转换。

url可以是一个字符串或者Request对象，Request存在的意义是便于在请求的时候传入一些信息，如果请求中需要加入headers（请求头）等信息，我们就可以利用更强大的Request类来构建一个请求

# 使用urlencode方法转换标准格式  
data = parse.urlencode(Form\_Data).encode('utf-8')

urllib.error异常

urllib.error可以接收有urllib.request产生的异常。urllib.error有两个方法，URLError和HTTPError。

URLError是OSError的一个子类，HTTPError是URLError的一个子类，服务器上HTTP的响应会返回一个状态码，根据这个HTTP状态码，我们可以知道我们的访问是否成功。

if \_\_name\_\_ == '\_\_main\_\_':  
 # URLError  
 url='http://www.iloveyou.com/' #一个不存在的连接  
 # HTTPError  
 url = 'http://www.douyu.com/Jack.html' #请求的资源没有在服务器上找 到,www.douyu.com这个服务器是存在的，但是我们要查找的Jack.html资源是没有的  
 req=request.Request(url)  
 try:  
 response=request.urlopen(req)  
 # html=response.read().decode('utf-8')  
 # print(html)  
 # except error.URLError as e:  
 # print(e.reason)  
 except error.HTTPError as e:  
 print(e.code)

URLError和HTTPError混合使用

如果想用HTTPError和URLError一起捕获异常，那么需要将HTTPError放在URLError的前面，因为HTTPError是URLError的一个子类。如果URLError放在前面，出现HTTP异常会先响应URLError，这样HTTPError就捕获不到错误信息了。

也可以使用hasattr函数判断URLError含有的属性，如果含有reason属性表明是URLError，如果含有code属性表明是HTTPError