**Python 3 多线程下载百度图片搜索结果**

下载简单页面

查看网页源码，发现同一张图片有四种网址：

"thumbURL": "http://img1.imgtn.bdimg.com/it/u=757023778,2840825931&fm=21&gp=0.jpg",

"middleURL": "http://img1.imgtn.bdimg.com/it/u=757023778,2840825931&fm=21&gp=0.jpg",

"hoverURL": "http://img1.imgtn.bdimg.com/it/u=757023778,2840825931&fm=23&gp=0.jpg",

"objURL": "http://imgsrc.baidu.com/forum/w=580/sign=b3bcc2f88a5494ee87220f111df4e0e1/78fed309b3de9c82913abac86a81800a18d84344.jpg",

经测试，前三种都有反爬虫措施，用浏览器可以打开，但是刷新一次就403 Forbidden。用爬虫获取不到图片。

第四种objURL是指图片的源网址，获取该网址会出现三种情况：

1. 正常。继续下载

2. 403 Forbidden。用continue跳过。

3. 出现异常。用try except处理。

代码如下：

import requests

import re

import os

url = r'http://image.baidu.com/search/index?tn=baiduimage&ipn=r&ct=201326592&cl=2&lm=-1

&st=-1&fm=detail&fr=&sf=1&fmq=1447473655189\_R&pv=&ic=0&nc=1&z=&se=&showtab=0&fb=0&width=&height=&face=0&istype=2&ie=utf-8&word=%E9%95%BF%E8%80%85%E8%9B%A4'

dirpath = r'F:\img'

html = requests.get(url).text

urls = re.findall(r'"objURL":"(.\*?)"', html)

if not os.path.isdir(dirpath):

os.mkdir(dirpath)

index = 1

for url in urls:

print("Downloading:", url)

try:

res = requests.get(url)

if str(res.status\_code)[0] == "4":

print("未下载成功：", url)

continue

except Exception as e:

print("未下载成功：", url)

filename = os.path.join(dirpath, str(index) + ".jpg")

with open(filename, 'wb') as f:

f.write(res.content)

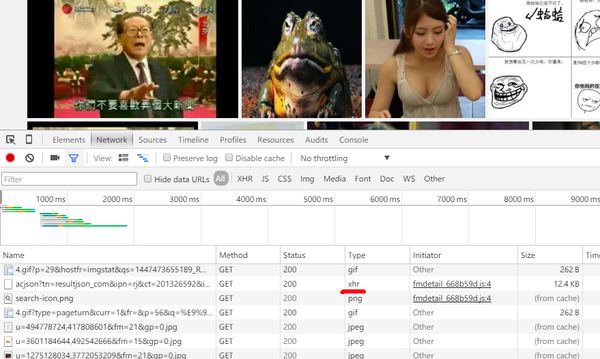
index += 1

print("下载结束，一共 %s 张图片" % index)

加载更多图片

但是上面的代码还有不足，当我们在网页中下拉时，百度会继续加载更多图片。需要再完善一下代码。

打开Chrome，按F12，切换到Network标签，然后将网页向下拉。这时浏览器地址栏的网址并没有改变，而网页中的图片却一张张增加，说明网页在后台与服务器交互数据。警察蜀黍，就是这家伙：

[](http://7xo8t2.com1.z0.glb.clouddn.com/img/%E7%BC%96%E7%A8%8B/Python/Python%203%20%E5%A4%9A%E7%BA%BF%E7%A8%8B%E4%B8%8B%E8%BD%BD%E7%99%BE%E5%BA%A6%E5%9B%BE%E7%89%87%E6%90%9C%E7%B4%A2%E7%BB%93%E6%9E%9CF12.png)

xhr全称XMLHttpRequest，详细介绍见百度：[XMLHTTPRequest\_百度百科](http://baike.baidu.com/link?url=59cVUdY8JuhiQ1NBfELSH93H_HFlTg9YoC5cqnjuyIvVHcyqx92fVNLTUWfvq8lesK5-h9HXgmbVuQseom0qu_" \t "_blank)

XHR英文全名XmlHttpRequest，中文可以解释为可扩展超文本传输请求。Xml可扩展标记语言，Http超文本传输协议，Request请求。XMLHttpRequest对象可以在不向服务器提交整个页面的情况下，实现局部更新网页。当页面全部加载完毕后，客户端通过该对象向服务器请求数据，服务器端接受数据并处理后，向客户端反馈数据。 XMLHttpRequest 对象提供了对 HTTP 协议的完全的访问，包括做出 POST 和 HEAD 请求以及普通的 GET 请求的能力。XMLHttpRequest 可以同步或异步返回 Web 服务器的响应，并且能以文本或者一个 DOM 文档形式返回内容。尽管名为 XMLHttpRequest，它并不限于和 XML 文档一起使用：它可以接收任何形式的[文本文档](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E6%96%87%E6%A1%A3" \t "_blank)。XMLHttpRequest 对象是名为 AJAX 的 Web 应用程序架构的一项关键功能。

点开看看：  
[](http://7xo8t2.com1.z0.glb.clouddn.com/img/%E7%BC%96%E7%A8%8B/Python/Python%203%20%E5%A4%9A%E7%BA%BF%E7%A8%8B%E4%B8%8B%E8%BD%BD%E7%99%BE%E5%BA%A6%E5%9B%BE%E7%89%87%E6%90%9C%E7%B4%A2%E7%BB%93%E6%9E%9CRequestURL.png)

这么长一串网址，没有头绪。下拉网页，再抓一个xhr，对比一下它们的Request URL，使用在线文字对比工具：[文本比较](http://home.putclub.com/tools/find/" \t "_blank)



URL末尾有三处变化，最后一项看上去是时间戳，经过测试，直接把它删了也没事。

那么只需要研究pn和gsm值。继续下拉，到底的时候点加载更多图片，多抓几个对比一下URL的末尾部分：

那么只需要研究pn和gsm值。继续下拉，到底的时候点加载更多图片，多抓几个对比一下URL的末尾部分：

pn=120&rn=60&gsm=78

pn=180&rn=60&gsm=b4

pn=240&rn=60&gsm=f0

pn=300&rn=60&gsm=12c

pn=360&rn=60&gsm=168

pn是一个60为步长的等差数列。gsm看上去是16进制，转换成十进制，发现它就是pn值，试了也可以删掉。

经测试，rn是步长值，最大只能取60，填入大于60的数，仍然以60为步长。如果删了rn，则步长值变为30。pn是图片编号，从0开始。

现在已经删了时间戳和gsm两项了，能不能让网址再短一点？继续观察，注意到：

&se=&tab=&width=&height=

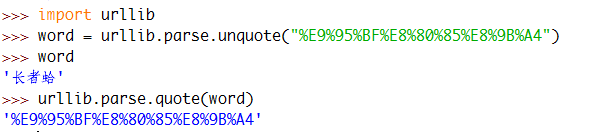
这几项没指定值，肯定没用，把没值的都删了。

再看看这两项：

queryWord=%E9%95%BF%E8%80%85%E8%9B%A4

word=%E9%95%BF%E8%80%85%E8%9B%A4

这就是我们本次搜索的关键词。网址中的中文会被编码成UTF-8，每个中文3个字节，每个字节前加上%号。编码和解码方法如下：



那么，我们可以写出指定关键词需要请求的所有网址：

import itertools

import urllib

def buildUrls(word):

word = urllib.parse.quote(word)

url = r"http://image.baidu.com/search/acjson?tn=resultjson\_com&ipn=rj&ct=201326592

&fp=result&queryWord={word}&cl=2&lm=-1&ie=utf-8&oe=utf-8&st=-1&ic=0&word={word}&face=0&istype=2nc=1&pn={pn}&rn=60"

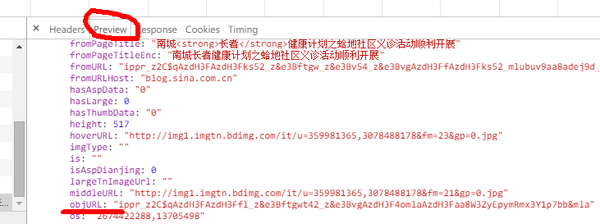
urls = (url.format(word=word, pn=x) for x in itertools.count(start=0, step=60))

return urls

上面的代码中，itertools.count(start=0, step=60)表示一个从0开始，60为步长值的无限等差数列。

把这个数列的数字分别填入url作为pn值，就得到一个无限容量的url生成器，注意生成器必须用圆括号，如果写成中括号就成了列表，程序会在这一步无限执行下去。

**下面开始解析每次获取的数据。我们点开看看，返回的是一串json数据。**



这个objURL怎么不是HTTP开头的。

试了几种方法都没成功，Google一下，找到这个：百度图片url解码

使用query词抓取百度图片数据的时候，为了简化抓取，使用firebug跟踪到百度图片请求返回了一个json格式数据，例如http://image.baidu.com/i?tn=resultjson\_com&ie=gbk&word=%B7%B6%B1%F9%B1%F9&cg=girl&pn=0&rn=60（该url已简化）。我们只需抓取这个json数据即可，但是这个json格式中的objectUrl和fromUrl却是加密了的数据，如ippr\_z2C$qAzdH3FAzdH3Fjgp\_z&e3Bvg6\_z&e3BvgAzdH3FstfpAzdH3Fda8candnAzdH3FWada8candnn98c9lndadl9\_z&e3B3r2。后又使用firebug发现，鼠标在点击连接时并没有向百度解密的请求，故判断解密程序在javascript中。

本来打算使用SpiderMonkey执行js来解密，但今天在网上偶然看到一段js解密url的代码，来自百度知道http://zhidao.baidu.com/link?url=APKpO\_ktmLa4QE0ZYlOqKyNdLul4rkDbV-LXBeDv40sHe0yDXS-LGc4A3ArNL2RoSY4xD2Z8M\_5vdmjm2nAOILhYWXE1ErtGMiPbi-paK4e，测试了下是对的，到这里这个问题总算搞定了。

解密方法很简单，秘钥是一个字符的对应关系，有2种映射：（1）多个字符映射为一个字符，'\_z2C$q'=>':','\_z&e3B'=>'.','AzdH3F'=>'/'，（2）单个字符映射为单字符。根据这个关系可将密文解密为明文，实现并不困难。

OK，既然明白了原理，我们写一个Python版的解密实现：

"""解码百度图片搜索json中传递的url

抓包可以获取加载更多图片时，服务器向网址传输的json。

其中originURL是特殊的字符串

解码前：

ippr\_z2C$qAzdH3FAzdH3Ffl\_z&e3Bftgwt42\_z&e3BvgAzdH3F4omlaAzdH3Faa8W3ZyEpymRmx3Y1p7bb&mla

解码后：

http://s9.sinaimg.cn/mw690/001WjZyEty6R6xjYdtu88&690

使用下面两张映射表进行解码。

"""

str\_table = {

'\_z2C$q': ':',

'\_z&e3B': '.',

'AzdH3F': '/'

}

char\_table = {

'w': 'a',

'k': 'b',

'v': 'c',

'1': 'd',

}

# str 的translate方法需要用单个字符的十进制unicode编码作为key

# value 中的数字会被当成十进制unicode编码转换成字符

# 也可以直接用字符串作为value

char\_table = {ord(key): ord(value) for key, value in char\_table.items()}

def decode(url):

# 先替换字符串

for key, value in str\_table.items():

url = url.replace(key, value)

# 再替换剩下的字符

return url.translate(char\_table)

if \_\_name\_\_ == '\_\_main\_\_':

url = r"ippr\_z2C$qAzdH3FAzdH3Ffl\_z&e3Bftgwt42\_z&e3BvgAzdH3F4omlaAzdH3Faa8W3Zy

EpymRmx3Y1p7bb&mla"

print(decode(url))

测试成功！

再从JSON字符串中找出所有的originURL：

re\_url = re.compile(r'"objURL":"(.\*?)"')  
imgs = re\_url.findall(html)

**单线程下载脚本**

整理一下流程：

* 生成网址列表
* 发送HTTP请求获取json数据
* 解析数据得到网址
* 下载

整合一下上面的代码，可以写出单线程的下载脚本：

import json

import itertools

import urllib

import requests

import os

import re

import sys

str\_table = {

'\_z2C$q': ':',

'\_z&e3B': '.',

'AzdH3F': '/'

}

char\_table = {

'w': 'a',

'k': 'b',

'v': 'c',

'1': 'd',

'j': 'e',

'u': 'f',

'2': 'g',

'i': 'h',

't': 'i',

'3': 'j',

'h': 'k',

's': 'l',

'4': 'm',

'g': 'n',

'5': 'o',

'r': 'p',

'q': 'q',

'6': 'r',

'f': 's',

'p': 't',

'7': 'u',

'e': 'v',

'o': 'w',

'8': '1',

'd': '2',

'n': '3',

'9': '4',

'c': '5',

'm': '6',

'0': '7',

'b': '8',

'l': '9',

'a': '0'

}

# str 的translate方法需要用单个字符的十进制unicode编码作为key

# value 中的数字会被当成十进制unicode编码转换成字符

# 也可以直接用字符串作为value

char\_table = {ord(key): ord(value) for key, value in char\_table.items()}

# 解码图片URL

def decode(url):

# 先替换字符串

for key, value in str\_table.items():

url = url.replace(key, value)

# 再替换剩下的字符

return url.translate(char\_table)

# 生成网址列表

def buildUrls(word):

word = urllib.parse.quote(word)

url = r"http://image.baidu.com/search/acjson?tn=resultjson\_com&ipn=rj&ct=201326592&fp=result&queryWord={word}&cl=2&lm=-1&ie=utf-8&oe=utf-8&st=-1&ic=0&word={word}&face=0&istype=2nc=1&pn={pn}&rn=60"

urls = (url.format(word=word, pn=x) for x in itertools.count(start=0, step=60))

return urls

# 解析JSON获取图片URL

re\_url = re.compile(r'"objURL":"(.\*?)"')

def resolveImgUrl(html):

imgUrls = [decode(x) for x in re\_url.findall(html)]

return imgUrls

def downImg(imgUrl, dirpath, imgName):

filename = os.path.join(dirpath, imgName)

try:

res = requests.get(imgUrl, timeout=15)

if str(res.status\_code)[0] == "4":

print(str(res.status\_code), ":" , imgUrl)

return False

except Exception as e:

print("抛出异常：", imgUrl)

print(e)

return False

with open(filename, "wb") as f:

f.write(res.content)

return True

def mkDir(dirName):

dirpath = os.path.join(sys.path[0], dirName)

if not os.path.exists(dirpath):

os.mkdir(dirpath)

return dirpath

if \_\_name\_\_ == '\_\_main\_\_':

print("欢迎使用百度图片下载脚本！\n目前仅支持单个关键词。")

print("下载结果保存在脚本目录下的results文件夹中。")

print("=" \* 50)

word = input("请输入你要下载的图片关键词：\n")

dirpath = mkDir("results")

urls = buildUrls(word)

index = 0

for url in urls:

print("正在请求：", url)

html = requests.get(url, timeout=10).content.decode('utf-8')

imgUrls = resolveImgUrl(html)

if len(imgUrls) == 0: # 没有图片则结束

break

for url in imgUrls:

if downImg(url, dirpath, str(index) + ".jpg"):

index += 1

print("已下载 %s 张" % index)

执行脚本：



## 多线程下载脚本

上面的代码仍然有改进空间：

从JSON数据看，该关键词相关的图片有一千多张，单线程下载太慢了，时间都花在网络和硬盘IO上。加上多线程可以大大提升效率。

既然1中已经获取到图片总数，那么网址的无限容量生成器可以改成list，方便添加多线程。

多线程一直没学好，想不到更优雅的写法，大家将就看一下吧，欢迎提出改进建议。

百度图片下载脚本之多线程版：

**#!/usr/bin/env python**

**# -\*- coding: utf-8 -\*-**

**# @Author: loveNight**

**# @Date: 2015-10-28 19:59:24**

**# @Last Modified by: loveNight**

**# @Last Modified time: 2015-11-15 19:24:57**

**import urllib**

**import requests**

**import os**

**import re**

**import sys**

**import time**

**import threading**

**from datetime import datetime as dt**

**from multiprocessing.dummy import Pool**

**from multiprocessing import Queue**

**class BaiduImgDownloader(object):**

**"""百度图片下载工具，目前只支持单个关键词"""**

**# 解码网址用的映射表**

**str\_table = {**

**'\_z2C$q': ':',**

**'\_z&e3B': '.',**

**'AzdH3F': '/'**

**}**

**char\_table = {**

**'w': 'a',**

**'k': 'b',**

**'v': 'c',**

**'1': 'd',**

**'j': 'e',**

**'u': 'f',**

**'2': 'g',**

**'i': 'h',**

**'t': 'i',**

**'3': 'j',**

**'h': 'k',**

**'s': 'l',**

**'4': 'm',**

**'g': 'n',**

**'5': 'o',**

**'r': 'p',**

**'q': 'q',**

**'6': 'r',**

**'f': 's',**

**'p': 't',**

**'7': 'u',**

**'e': 'v',**

**'o': 'w',**

**'8': '1',**

**'d': '2',**

**'n': '3',**

**'9': '4',**

**'c': '5',**

**'m': '6',**

**'0': '7',**

**'b': '8',**

**'l': '9',**

**'a': '0'**

**}**

**re\_objURL = re.compile(r'"objURL":"(.\*?)".\*?"type":"(.\*?)"')**

**re\_downNum = re.compile(r"已下载\s(\d+)\s张图片")**

**headers = {**

**"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36",**

**"Accept-Encoding": "gzip, deflate, sdch",**

**}**

**def \_\_init\_\_(self, word, dirpath=None, processNum=30):**

**if " " in word:**

**raise AttributeError("本脚本仅支持单个关键字")**

**self.word = word**

**self.char\_table = {ord(key): ord(value)**

**for key, value in BaiduImgDownloader.char\_table.items()}**

**if not dirpath:**

**dirpath = os.path.join(sys.path[0], 'results')**

**self.dirpath = dirpath**

**self.jsonUrlFile = os.path.join(sys.path[0], 'jsonUrl.txt')**

**self.logFile = os.path.join(sys.path[0], 'logInfo.txt')**

**self.errorFile = os.path.join(sys.path[0], 'errorUrl.txt')**

**if os.path.exists(self.errorFile):**

**os.remove(self.errorFile)**

**if not os.path.exists(self.dirpath):**

**os.mkdir(self.dirpath)**

**self.pool = Pool(30)**

**self.session = requests.Session()**

**self.session.headers = BaiduImgDownloader.headers**

**self.queue = Queue()**

**self.messageQueue = Queue()**

**self.index = 0 # 图片起始编号，牵涉到计数，不要更改**

**self.promptNum = 10 # 下载几张图片提示一次**

**self.lock = threading.Lock()**

**self.delay = 1.5 # 网络请求太频繁会被封**

**self.QUIT = "QUIT" # Queue中表示任务结束**

**self.printPrefix = "\*\*" # 用于指定在控制台输出**

**def start(self):**

**# 控制台输出线程**

**t = threading.Thread(target=self.\_\_log)**

**t.setDaemon(True)**

**t.start()**

**self.messageQueue.put(self.printPrefix + "脚本开始执行")**

**start\_time = dt.now()**

**urls = self.\_\_buildUrls()**

**self.messageQueue.put(self.printPrefix + "已获取 %s 个Json请求网址" % len(urls))**

**# 解析出所有图片网址，该方法会阻塞直到任务完成**

**self.pool.map(self.\_\_resolveImgUrl, urls)**

**while self.queue.qsize():**

**imgs = self.queue.get()**

**self.pool.map\_async(self.\_\_downImg, imgs)**

**self.pool.close()**

**self.pool.join()**

**self.messageQueue.put(self.printPrefix + "下载完成！已下载 %s 张图片，总用时 %s" %**

**(self.index, dt.now() - start\_time))**

**self.messageQueue.put(self.printPrefix + "请到 %s 查看结果！" % self.dirpath)**

**self.messageQueue.put(self.printPrefix + "错误信息保存在 %s" % self.errorFile)**

**self.messageQueue.put(self.QUIT)**

**def \_\_log(self):**

**"""控制台输出，加锁以免被多线程打乱"""**

**with open(self.logFile, "w", encoding = "utf-8") as f:**

**while True:**

**message = self.messageQueue.get()**

**if message == self.QUIT:**

**break**

**message = str(dt.now()) + " " + message**

**if self.printPrefix in message:**

**print(message)**

**elif "已下载" in message:**

**# 下载N张图片提示一次**

**downNum = self.re\_downNum.findall(message)**

**if downNum and int(downNum[0]) % self.promptNum == 0:**

**print(message)**

**f.write(message + '\n')**

**f.flush()**

**def \_\_getIndex(self):**

**"""获取文件编号"""**

**self.lock.acquire()**

**try:**

**return self.index**

**finally:**

**self.index += 1**

**self.lock.release()**

**def decode(self, url):**

**"""解码图片URL**

**解码前：**

**ippr\_z2C$qAzdH3FAzdH3Ffl\_z&e3Bftgwt42\_z&e3BvgAzdH3F4omlaAzdH3Faa8W3ZyEpymRmx3Y1p7bb&mla**

**解码后：**

**http://s9.sinaimg.cn/mw690/001WjZyEty6R6xjYdtu88&690**

**"""**

**# 先替换字符串**

**for key, value in self.str\_table.items():**

**url = url.replace(key, value)**

**# 再替换剩下的字符**

**return url.translate(self.char\_table)**

**def \_\_buildUrls(self):**

**"""json请求网址生成器"""**

**word = urllib.parse.quote(self.word)**

**url = r"http://image.baidu.com/search/acjson?tn=resultjson\_com&ipn=rj&ct=201326592&fp=result&queryWord={word}&cl=2&lm=-1&ie=utf-8&oe=utf-8&st=-1&ic=0&word={word}&face=0&istype=2nc=1&pn={pn}&rn=60"**

**time.sleep(self.delay)**

**html = self.session.get(url.format(word=word, pn=0), timeout = 15).content.decode('utf-8')**

**results = re.findall(r'"displayNum":(\d+),', html)**

**maxNum = int(results[0]) if results else 0**

**urls = [url.format(word=word, pn=x)**

**for x in range(0, maxNum + 1, 60)]**

**with open(self.jsonUrlFile, "w", encoding="utf-8") as f:**

**for url in urls:**

**f.write(url + "\n")**

**return urls**

**def \_\_resolveImgUrl(self, url):**

**"""从指定网页中解析出图片URL"""**

**time.sleep(self.delay)**

**html = self.session.get(url, timeout = 15).content.decode('utf-8')**

**datas = self.re\_objURL.findall(html)**

**imgs = [Image(self.decode(x[0]), x[1]) for x in datas]**

**self.messageQueue.put(self.printPrefix + "已解析出 %s 个图片网址" % len(imgs))**

**self.queue.put(imgs)**

**def \_\_downImg(self, img):**

**"""下载单张图片，传入的是Image对象"""**

**imgUrl = img.url**

**# self.messageQueue.put("线程 %s 正在下载 %s " %**

**# (threading.current\_thread().name, imgUrl))**

**try:**

**time.sleep(self.delay)**

**res = self.session.get(imgUrl, timeout = 15)**

**message = None**

**if str(res.status\_code)[0] == "4":**

**message = "\n%s： %s" % (res.status\_code, imgUrl)**

**elif "text/html" in res.headers["Content-Type"]:**

**message = "\n无法打开图片： %s" % imgUrl**

**except Exception as e:**

**message = "\n抛出异常： %s\n%s" % (imgUrl, str(e))**

**finally:**

**if message:**

**self.messageQueue.put(message)**

**self.\_\_saveError(message)**

**return**

**index = self.\_\_getIndex()**

**# index从0开始**

**self.messageQueue.put("已下载 %s 张图片：%s" % (index + 1, imgUrl))**

**filename = os.path.join(self.dirpath, str(index) + "." + img.type)**

**with open(filename, "wb") as f:**

**f.write(res.content)**

**def \_\_saveError(self, message):**

**self.lock.acquire()**

**try:**

**with open(self.errorFile, "a", encoding="utf-8") as f:**

**f.write(message)**

**finally:**

**self.lock.release()**

**class Image(object):**

**"""图片类，保存图片信息"""**

**def \_\_init\_\_(self, url, type):**

**super(Image, self).\_\_init\_\_()**

**self.url = url**

**self.type = type**

**if \_\_name\_\_ == '\_\_main\_\_':**

**print("欢迎使用百度图片下载脚本！\n目前仅支持单个关键词。")**

**print("下载结果保存在脚本目录下的results文件夹中。")**

**print("=" \* 50)**

**word = input("请输入你要下载的图片关键词：\n")**

**down = BaiduImgDownloader(word)**

**down.start()**

执行脚本：

**欢迎使用百度图片下载脚本！**

**目前仅支持单个关键词。**

下载结果保存在脚本目录下的results文件夹中。

==================================================

请输入你要下载的图片关键词：

长者蛤

2015-11-15 19:25:11.726878 \*\*脚本开始执行

2015-11-15 19:25:16.292022 \*\*已获取 20 个Json请求网址

2015-11-15 19:25:17.885767 \*\*已解析出 30 个图片网址

2015-11-15 19:25:17.917020 \*\*已解析出 60 个图片网址

.....

.....中间省略

.....

2015-11-15 19:33:31.726739 已下载 980 张图片：http://bbs.nantaihu.com/bbs/UpImages2008/2010/8/3/U97946\_201083171218512-2.jpg

2015-11-15 19:33:32.695518 已下载 990 张图片：http://pf.u.51img1.com/f9/7f/huangbaoling0\_180.gif?v=20120224161006

2015-11-15 19:33:45.473957 已下载 1000 张图片：http://library.taiwanschoolnet.org/cyberfair2003/C0312970394/narrative.files/image018.jpg

2015-11-15 19:33:50.749544 \*\*下载完成！已下载 1000 张图片，总用时 0:08:39.022666

2015-11-15 19:33:50.765169 \*\*请到 F:\PythonWorkspace\Learn\results 查看结果！

2015-11-15 19:33:50.858880 \*\*错误信息保存在 F:\PythonWorkspace\Learn\errorUrl.txt

**仍然可以改进的地方：**

**如果要搜索多个关键词，buildUrls方法应该怎么改？**

**如果脚本中途意外结束（比如被熊孩子点了X），如何继续下载？**

**线程池中的线程数需要多次测试才能找到最优值。**